Introduction à la cybersécurité

Université Sorbonne Paris Nord - Master 2

iro@cryptosec.org | @secucrypt

Novembre 2020

Version 5

Contenu

~~~

Les principes généraux importants et transverses sont encadrés tout au long du cours, et récapitulés en conclusion

Les bases de la cryptographie ayant été vues, ce cours ne comporte pas de chapitre dédié à ces techniques. En cas de besoin de quelques rappels :

http://cryptosec.org/?Introduction-to-Cryptography

https://marumari.github.io/crypto-presentation/#/1
~~~

l’auteur initial vous soutient ou soutient la façon dont vous avez utilisé le présent document), ***pas d’utilisation commerciale*** *(vous n'êtes pas autorisé à faire un usage commercial de ce document, tout ou partie du matériel le composant),* ***partage dans les mêmes conditions*** *(dans le cas où vous effectuez un remix, que vous transformez, ou créez à partir du matériel composant le document original, vous devez diffuser le document modifié dans les même conditions, c'est-à-dire avec la même licence avec laquelle le document original a été diffusé),* ***pas de restrictions complémentaires*** *(vous n'êtes pas autorisé à appliquer des conditions légales ou des mesures techniques qui restreindraient légalement autrui à utiliser le document dans les conditions décrites par la licence).*

Ce qui précède est un résumé (et non pas un substitut) de la licence : https://creativecommons.org/licenses/by-nc-sa/2.0/fr/legalcode

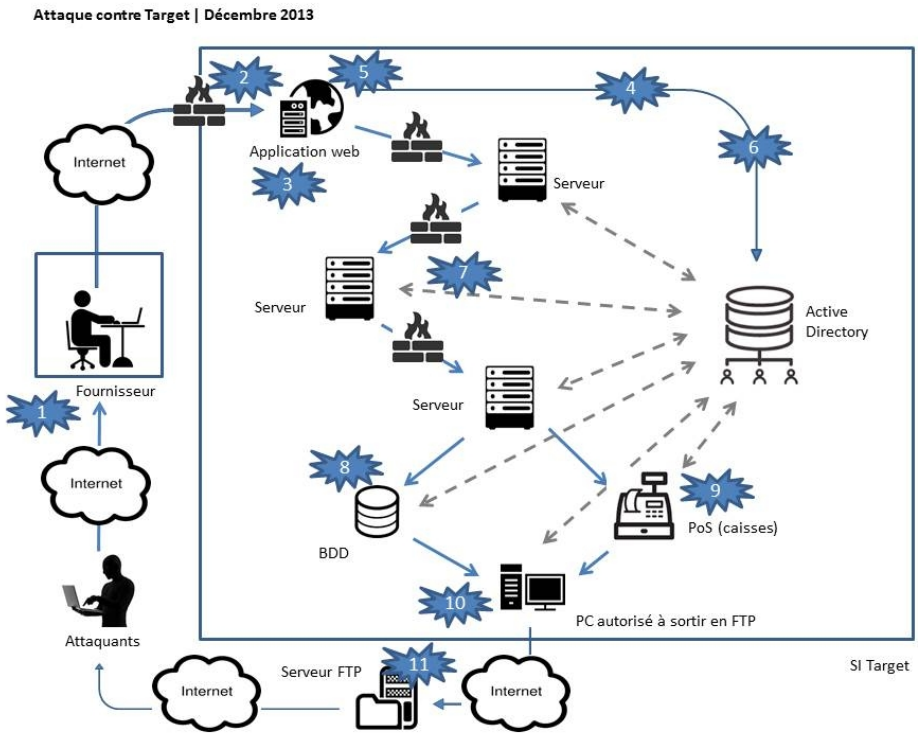
I. Une attaque réelle (1/2)

Target, une grande chaine de supermarchés étasunienne – Décembre 2013

Vol de 40 millions de numéros de carte de crédit et des données personnelles de 70 millions de personnes.

Dommages estimés : plus de 150 millions de dollars (mais les estimations de pertes financières sont toujours sujettes à caution).

Que s'est-il passé ?



Toutes les informations ne sont pas certaines. Mais l'essentiel a dû se passer à peu près comme ça :

1. Installation d'un *malware* (Citadel, introduit via *spear phishing* – message de *phishing,* mais très ciblé et personnalisé) qui vole des données de connexion depuis un PC d'un sous-traitant gérant les installations d'air conditionné. *Cela commence par une défaillance humaine ;*
2. Connexion chez Target avec les identifiants. Accès à l'application web pour le sous-traitant (facturation, gestion projet, gestion des contrats. *Point notable : il s'agissait d'une application anodine, non critique, sans accès aux systèmes visés*) ;
3. Exploitation d'une vulnérabilité de l'interface web, qui permet d'uploader du code sur le serveur (*upload* d'un webshell PHP) ;
4. Reconnaissance : recherche de cibles (serveurs contenant des numéros de CB, bases de données) pour propagation en requêtant l'annuaire Active Directory (en LDAP / recherche string "MSSQLSvc") ;
5. Vol d'un jeton d'authentification (*Pass-the-Hash*) dans la mémoire de l'application web ;
6. Création d'un nouveau compte admin de domaine dans l'AD en utilisant le jeton d'authentification volé (net user / group) ;
7. Propagation vers les machines cibles avec les comptes admin de domaine (RDP et PsExe pour exécution d'autres softs à distance, scan IP pour accès réseau). *Utilisation d'outils d'admin légitimes, non détectés ;*
8. Vol de données personnelles, extraites de bases de données, via SQL (pas de CBs). *Target était conforme PCI-DSS : pas de données cartes dans leurs bases de données ; les attaquants ont changé de stratégie : ils ont ciblé les PoS (Points of Sale) ;*
9. Installation du *malware* Kaptoxa sur les caisses enregistreuses (PoS), vol de données 40 millions cartes de crédit. *Le seul vrai outil de hacking ;*
10. Envoi des données volées via des partages vers une machine pouvant sortir en FTP ;
11. Exfiltration via FTP.

Pendant que l'attaque était en cours, l'équipe qui monitorait la sécurité du système d'information de Target depuis Bangalore, en Inde, a détecté quelque chose et alerté les équipes opérationnelles à Minneapolis. Mais l'alerte est restée lettre morte, aucune action n'a été entreprise.

Cette conclusion introduit un premier principe, essentiel :

> La sécurité n'est jamais un logiciel, un produit ; la sécurité ce sont des processus.

La complexité et l'inventivité des attaquants mènent au constat suivant :

> À mesure que la sophistication des usages croit, celle des malveillances aussi.

Question	Qu'est-ce qui aurait permis d'identifier et peut-être de réduire ces risques ?
Réponse	• Des audits sur site, chez le (les) prestataire(s) ; • Des tests d'ingénierie sociale ; • Des tests de sécurité, *pentests (penetration tests)* ; • Des scans ; • Des audits de code ; • Des exercices et tests des processus de détection ; • Des exercices d'entraînement à la gestion des situations de crise…

II. Une attaque réelle (2/2)

Avant l'élection US de 2016, trois experts néerlandais avaient réussi à s'authentifier sur le compte Twitter de Trump. Comment ? En cherchant l'intéressé dans une base de comptes ayant fuité en 2012, provenant de LinkedIn. La base ne contenait pas les mots de passe, mais leurs condensés. (https://www.theregister.com/2020/09/11/trump_twitter_account_recycled_password/)

Le condensé associé à Trump était :

07b8938319c267dcdb501665220204bbde87bf1d

Que faire avec ça ? Comment retrouver le mot de passe correspondant ?

Ils ont utilisé John the Ripper (https://www.openwall.com/john/) pour trouver le mot de passe.

Mais dans ce cas, parce que les mots de passes n'étaient pas salés, ils auraient aussi pu utiliser des *rainbow table*, comme celles accessibles sur https://crackstation.net/

(pour vérifier, vous pouvez taper dans un terminal : echo -n yourfired | sha1sum)

Question	Quelles erreurs a commis Trump et son équipe ?
Réponse	• Mot de passe très faible ; • Utilisation d'un même mot de passe sur plusieurs sites ; • Pas d'authentification double facteur ; • Mot de passe inchangé pendant des années, y compris après un *hack ;* • Condensé non salé ; • Les autorités US n'ont pas donné suite au signalement…

Pour vos comptes, vérifiez régulièrement https://haveibeenpwned.com/

III. La sécurité et les risques

La sécurité est une situation objective caractérisée par la seule présence de risques maitrisés.

En principe la *sûreté* est relative aux risques accidentels (résultant éventuellement d'actions humaines involontaires), tandis que la *sécurité* traite des actes malveillants.

Ainsi, la sûreté traite des mécanismes participant à la continuité du fonctionnement d'un système, tandis que la sécurité traite des mécanismes protégeant l'information des accès et manipulations non autorisés.

Mais dans le domaine informatique, un seul terme est usuellement utilisé, « sécurité », tandis que « sûreté » est souvent utilisé dans le domaine de la sécurité physique. À noter qu'avec l'informatisation croissante des mécanismes de sécurité physique, la frontière entre ces deux domaines devient de plus en plus poreuse.

(Black Hat 2016 – Las Vegas)

Les risques sont des contingences indésirables qui peuvent être avérées, potentielles ou futures. Ils découlent d'une probabilité qu'une menace exploite, intentionnellement ou non, une vulnérabilité.

Toute chose, en particulier un individu ou une organisation, s'efforce de persévérer dans son être (Baruch Spinoza, Éthique III, Proposition VI, 1677).

Elle va donc chercher à minimiser les contingences adverses. C'est-à-dire qu'elle va essayer d'évoluer, dans l'espace et le temps, entre des états où elle maitrise les risques qui la menacent.

Ce que l'on voit, c'est que la sécurité, élément essentiel de l'existence, résulte d'une dynamique de maitrise.

Cette dynamique offre quatre manières de **modifier un risque** unitaire :

- Tenter de le **réduire** : cela consiste soit à diminuer la probabilité que se matérialise une menace, soit à réduire son impact, ses conséquences, s'il se réalise ;
- L'**accepter** : cela consiste à considérer que l'effet indésirable du risque, qu'il soit faible ou létal, est acceptable au regard des enjeux, des objectifs, du désir ou des obligations de l'individu ou de l'organisation ;
- Le **refuser** : cela signifie modifier sa propre essence, ce que l'on est ou ce que l'on fait afin de se soustraire à des vulnérabilités. Ce qui revient à ne pas réaliser une action ou amputer une fonctionnalité ;
- Le **transférer** : ce qui consiste à le faire peser sur un tiers, que ce déplacement soit consenti ou non (exemple des assureurs ; à noter

cependant que le transfert contractuel – faire porter la responsabilité d'une attaque sur un sous-traitant par exemple – est aléatoire et souvent théorique. Dans la plupart des cas, le donneur d'ordre conserve une part de responsabilité).

La sécurisation résulte toujours d'une dynamique, même lorsque le système à sécuriser est parfaitement statique et n'évolue pas. D'une part parce que les menaces exogènes, elles, peuvent évoluer, et d'autre part parce que la réaction, la diminution des impacts d'un incident, sera nécessairement toujours dynamique.

De cette caractéristique dynamique on peut enfin déduire que la sécurité relève de *processus* impliquant l'individu ou l'organisation, et non uniquement d'une méthodologie, d'un système, d'une expertise ou d'un produit.

Un élément essentiel et constitutif des risques et de la sécurité est l'imprévu.

Par définition, les contingences indésirables que sont les risques ne sont pas déterministes. S'ils le sont, alors il s'agit de contraintes. C'est-à-dire qu'au cœur de toute action et de toute réflexion autour de la sécurité se niche l'incertain à venir, l'indéterminé, le hasard, la probabilité.

Ainsi, nombre d'experts et d'équipes sécurité travaillent presque en aveugle, cherchant en particulier à empêcher le syndrome du « cygne noir » (cf. la « théorie » du *cygne noir*, une métaphore qui décrit des évènements imprévisibles de faible probabilité d'occurrence, mais qui, s'ils se réalisent, ont des conséquences d'une portée exceptionnelle).

Une façon classique de présenter les risques :

Risque = (Menace) x (Vulnérabilité)

Quantification du risque = (Impact) x (Probabilité d'occurrence)

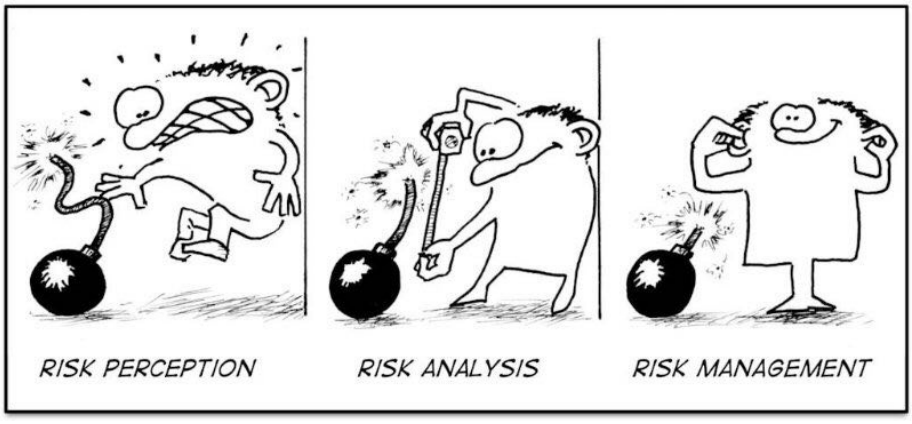
Il s'agit alors d'une logique combinatoire se matérialisant par des matrices Impact x Probabilité dont les cases vont être vertes, jeunes, oranges, rouges…

Exemple de matrice d'estimation de risques :

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

(https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology)

Question	Quel est le problème de cette approche de quantification des risques ?
Réponse	La difficile estimation des probabilités si l'on ne dispose pas de statistiques fiables, ce qui n'est pas le cas concernant les attaques informatiques.



Les impacts d'un risque qui se matérialise peuvent se décliner en trois catégories :

- **Intégrité** ;

- **Confidentialité** ;
- **Disponibilité**.

Les mesures de sécurité peuvent elles aussi se diviser en trois catégories :

- **Prévention** : ce sont les mesures de sécurité que l'on met en place pour empêcher la matérialisation d'un risque, empêcher ou retarder les actions d'attaquants. Elles réduisent la probabilité d'occurrence, souvent liée à la surface d'attaque ;
- **Détection** : nécessaires si l'on admet qu'aucune mesure de sécurité préventive ne peut arrêter un attaquant suffisamment motivé, compétent ou doté de moyens assez importants ;
- **Réaction** : ces mesures (cf. l'aspect « dynamique » évoqué plus haut) visent à réduire l'impact d'une attaque, suite à sa détection.

IV. Équilibre entre la sécurité et les usages

Avant de poursuivre, il est important de souligner quelque chose qui servira de fil rouge tout au long de ce cours : il est essentiel de toujours chercher un bon équilibre entre les usages, la liberté des utilisateurs et la sécurité.

Le risque 0 n'est envisageable qu'en l'absence de vie et d'activité. Toute action comportera toujours des risques.

De plus, si la sécurité impose trop de contraintes il est probable que les utilisateurs trouveront des chemins de traverse entrainant des risques plus élevés (exemple d'une politique de mots de passe trop sévère qui incite les utilisateurs à les noter sur des post-it…).

La sécurité a un cout, en énergie, en temps, en argent, mais aussi en usages et en liberté.

Il faut toujours se demander si le cout de la sécurité n'est pas disproportionné comparé à ce qu'elle protège, ou ne met pas en péril certains principes essentiels.

La bonne sécurité résulte toujours d'un équilibre.

V. Paysage des menaces

Pour faire de la bonne sécurité, il faut savoir penser comme un attaquant.

Et donc essayer de savoir qui ils peuvent être, quelles sont leurs motivations, leurs moyens…

Cyberdélinquance

Objectifs : Gains financiers

Moyens : vol et revente de données, fraudes et détournements de fonds, demandes de rançons, mise à disposition d'infrastructures d'attaques (Crimeware as a Service – CaaS, botnets, etc.)

> **Member of the JIGSAW group:** We are hired by corporation to cyber disrupt day-to-day business of their competition. I don't even know how you got it.
>
> **F-Secure:** Interesting. So that's why the ransom is so low - because you are already getting paid by the corporation, so you are mostly interested in disrupting the business rather than making a lot of money off the ransom? Is it like a legitimate corporation, and is it well-known?
>
> **Member of the JIGSAW group:** Yes, big name corporation. Fortune 500 company. The purpose was just to lock files to delay a corporation's production time to allow our clients to introduce a similar product into the market first.

(exemple de dialogue entre un gang diffusant des *ransomware* et la société F-Secure – Juillet 2016)

sign in become a supporter subscribe search jobs dating more International

theguardian

UK world sport football opinion culture business lifestyle fashion environment tech travel browse all sections

home › tech

Cybercrime
The Observer

City banks plan to hoard bitcoins to help them pay cyber ransoms

Experts say blue chip companies have decided it's cheaper to deal with extortionists than risk damaging attacks

Jamie Doward

Saturday 22 October 2016 22.30 BST

This article is 1 month old

170

Criminal gangs are said to prefer payment in bitcoins because the virtual currency cannot be traced. Photograph: George Frey/Getty Images

Several of London's largest banks are looking to stockpile bitcoins in order to pay off cyber criminals who threaten to bring down their critical IT systems.

The virtual currency, which is highly prized by criminal networks because it is

Most popular

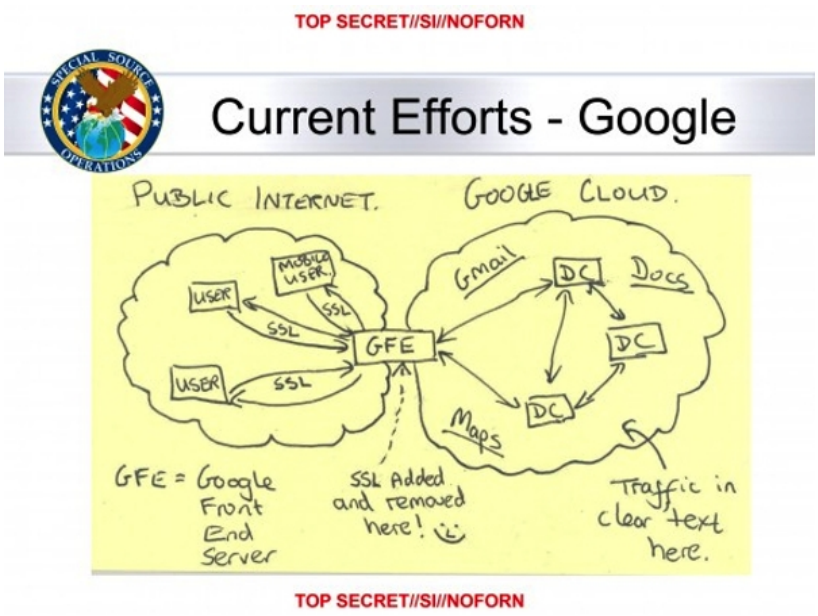
Autumn statement: OBR says Brexit impact could be even worse than feared - live

Script kiddies

Objectifs : Divers / Opportunistes

Moyens : Opportuniste, utilisation d'outils et de services d'attaques

États et groupes paraétatiques



Objectifs : Espionnage, désinformation, cyberguerre

Moyens parfois très importants : dénis de services, attaques ciblées, portes dérobées…

(TS//SI//NF) Left: Intercepted packages are opened carefully; Right: A "load station" implants a beacon

(photo de l'unité *Tailored Access Operations* de la NSA – Révélations d'Edward Snowden, juin 2013)

Concurrence

Objectifs : Espionnage industriel, avantage concurrentiel

Moyens : Dénis de services, vol de données, attaques ciblées, ransomwares, via des mercenaires

Malveillances internes

Objectifs : Divers

Moyens : Accès à des ressources non protégées, exploitation de lacunes de sécurité

Cyberprotestations

Objectifs : Impact médiatique, politique

Moyens : *Defacements*, dénis de service, vol de données

On voit que tous ces types d'attaquants vont déployer divers types d'attaques :

- **Attaques « au chalut »,** via des campagnes massives de courriels piégés (*phishing*), des sites compromis contenant des *malware*s, de faux sites pour récupérer des identifiants / mots de passe, attaques dites du « point d'eau »…
  Exemple : tous les "scams" et malwares de phishing que vous recevez, non personalisés
- **Attaques ciblées** avec une dimension d'ingénierie sociale, par l'envoi de courriels personnalisés (*spear phishing*), de clés USB piégées, via la mise en place de faux sites personnalisés, coups de téléphone (exemple : fraudes au président)…
  Exemple : En 2012, dans le cadre du G20, contre les banques centrales; en 2012, contre l'Elysée, utilisation de "Flame" et FB...
- **Intrusions** en ciblant des défauts de protection, des vulnérabilités applicatives, des données mal protégées…
  Exemple : 2016 contre la Banque centrale du Bangladesh
- ***Supply chain attacks*** : les attaquants compromettent un service en charge de la distribution de logiciels ou de mises à jour. Via ce pivot, ils sont ensuite capables de distribuer des logiciels malveillants à leurs cibles.
- **Dénis de service** (DoS) et Dénis de service distribués (DdoS)
  Exemple : Dyn (provider de DNS), en 2016: botnet IoT, plusieurs centaines de Gbps
- **Distribution de matériel ou logiciels piégés** (portes dérobées – *backdoors*…)
  Exemple : Stuxnet, 2010, ou les *skimmers*

- …

Il existe beaucoup de ressources d’accès libre qui permettent d’obtenir des informations sur des systèmes, par exemple : https://www.threatcrowd.org ou https://www.virustotal.com.

VI. Défense périmétrique

La défense périmétrique, ce sont toutes les mesures de sécurité visant à isoler un système d'information de « l'extérieur ».

Le concept embarque des mesures et dispositifs permettant par exemple de :

- Dissimuler le réseau et les services internes vis-à-vis de l'extérieur ;
- Différencier et filtrer le trafic entrant et sortant, indépendamment de ce qu'il contient ;
- Filtrer les données entrantes, c'est-à-dire le contenu du trafic ;
- Délimiter des zones de confiance variable: réseau local, DMZ externe ;
- Détecter des tentatives d'intrusion ;
- Créer des points d'accès distants (pour connexions à d'autres réseaux, à Internet).

Remarque : Avec la multiplication des applications web, la virtualisation, la diversité des terminaux, la coexistence des usages professionnels et privés, les accès distants, le « cloud »… il est de plus en plus difficile de distinguer « l'intérieur » de « l'extérieur ».

Filtrage réseau

Pare-feu

Un pare-feu est un équipement de sécurité réseau qui permet de filtrer et surveiller le trafic entre des zones réseau, selon des règles préétablies.

Ils permettent de séparer une zone réseau de confiance (comme le LAN d'un SI) d'une zone non sure, comme Internet.

(girafe anonyme)

Question	Imaginons que vous soyez en voiture et que vous deviez rejoindre un village, pour y acheter un cercle. Puis vous devrez en revenir (avec le cercle dans le coffre de votre voiture). À l'approche du village, une girafe persuadée de l'importance des frontières est plantée au milieu de la route et contrôle ceux qui veulent passer. Selon quels principes peut-elle agir pour vous laisser passer ou non, à l'aller et au retour ? (si la girafe se comportait comme un pare-feu)
Réponse	- Elle peut obéir à une **règle** qui, selon votre lieu de provenance et votre destination déclarée, ou la couleur de votre voiture, vous laisse passer ou non. À votre retour, une **autre règle** devra lui dire si elle doit vous laisser passer ou non. Elle se comporterait alors comme un **pare-feu sans état**. - Elle peut, comme précédemment, obéir à **une règle** pour vous laisser passer. Par contre, au retour, elle se **souviendra** de vous et vous laissera passer. Elle se comporterait alors comme un **pare-feu à états.** - Aux comportements précédents, la girafe peut ajouter **des inspections**, comme par exemple vérifier à l'aller que vous avez bien de l'argent pour payer votre cercle et, au retour, vérifier que le cercle

	que vous avez acheté est bien un cercle et non un carré. Elle se comporterait alors comme **pare-feu avec inspection protocolaire.** - La girafe peut aussi ne pas se soucier d'où vous venez, ni de la couleur de votre voiture. Elle vous identifie, passe un coup de fil au magasin de cercles et vous laisse passer si quelqu'un vous connait et vous attend pour vous vendre un cercle. Elle se comporterait alors comme **pare-feu avec filtrage utilisateur**.

Les premiers pare-feu ne géraient **pas les états**.

Les pare-feu modernes sont dits « **à états** » (*stateful*) : ils conservent l'état des connexions (TCP, UDP… / @IP, ports, numéros de séquence des paquets…) et savent déterminer si un paquet initie une connexion ou fait partie d'une connexion existante.

Il existe deux sortes de pare-feu : les pare-feu réseau, qui sont souvent des *appliances*, et les pare-feu locaux (*host-based firewall*).

Ils filtrent au niveau des paquets, niveau 4 du modèle OSI.

Certains protocoles sont « à état », comme TCP (SYN, SYN-ACK, ACK), il sera facile pour le pare-feu de gérer ça.

Pour des protocoles comme UDP ou ICMP, la mémoire des connexions ne peut se faire que par la conservation de l'historique des adresses et des ports.

Un des grands avantages des pare-feu « à états » est que pour les connexions établies, les paquets ne doivent être examinés qu'en regardant les tables d'états des sessions, et non les règles de blocage ou d'autorisation des flux.

Pour les flux non autorisés, on a en général le choix entre rejet explicite (*drop*) et rejet silencieux (*reject*) :

- *reject* : une erreur est renvoyée à l'expéditeur du paquet (ICMP) ;
- *drop* : rejet, mais sans retour. Le *drop* peut causer des lenteurs côté client (attente de *timeout* en l'absence de retour).

Options de règles typiques :

- Tout ce qui n'est pas explicitement interdit est autorisé : liste noire (*blacklist*) ;
- Tout ce qui n'est pas explicitement autorisé est interdit : liste blanche (*whitelist*).

Aujourd'hui, les pare-feu savent aussi faire de **l'inspection protocolaire**, embarquent des modules de détection d'intrusion (IDS/IPS) et savent **identifier les utilisateurs** autrement que par les @IP / port.

Problèmes typiques que l'on peut rencontrer sur les pare-feu :

- Accumulation de règles contradictoires ;
- Absence d'audit ;
- Ouvertures temporaires (par exemple en cas d'incident), et non-fermeture ultérieure.

De plus :

Il est impossible d'interdire ce que vous devez autoriser. Sachez faire accepter les risques identifiés.

L'avenir de la segmentation réseau : probablement le concept de *Zero Trust Network*…

Dans votre vie de développeur, vous allez souvent dire : c'est la faute du pare-feu ! Analysez les flux avant de l'affirmer (les outils les plus courants sont Wireshark au niveau réseau, Burp Suite au niveau applicatif).

Une méthode empirique qui fonctionne souvent assez bien :

Il ne faut pas autoriser plus d'un ANY dans le triplet : (source, destination, protocole)

Exemples :

Autoriser le trafic HTTP du réseau interne (10.0.0.0/24) vers internet :

hostname(config)# access-list HTTP-ONLY extended permit tcp 10.0.0.0 255.255.255.0 any eq 80

hostname(config)# access-group HTTP-ONLY in interface inside

(il y a un DENY ALL à la fin de chaque ACL)

DMZ

Une « zone démilitarisée » est une zone entre deux zones de sensibilité différentes, par exemple un LAN et Internet.

Le terme est d'origine militaire et désigne des zones tampons entre deux territoires dont les armées ont conclu un armistice ou un traité de paix et où l'activité militaire est interdite ou très réduite. Exemples : En Rhénanie suite au traité de Versailles, entre la Corée du Nord et la Corée du Sud…

Elles peuvent contenir des services qui doivent être exposés sur Internet, comme des serveurs web, de messagerie, les *front-end* d'applications, sans autoriser des flux entrants jusqu'au cœur du réseau.

Elles peuvent également héberger des dispositifs de sécurité comme des passerelles VPN, WAF ou systèmes d'authentification, des reverse-proxy pour traiter les flux dans une zone tampon avant de parvenir au réseau interne.

En général les DMZ sont construites sur un pare-feu assurant le filtrage des zones séparées.

Remarque : les DMZ sont dans la partie « périmétrique » parce qu'elles servent à faire l'interface entre des zones de confiance différente et que bien des fonctions de filtrage y sont souvent assurées.

Concernant les flux, un principe général (mais qui ne peut pas toujours être appliqué) est :

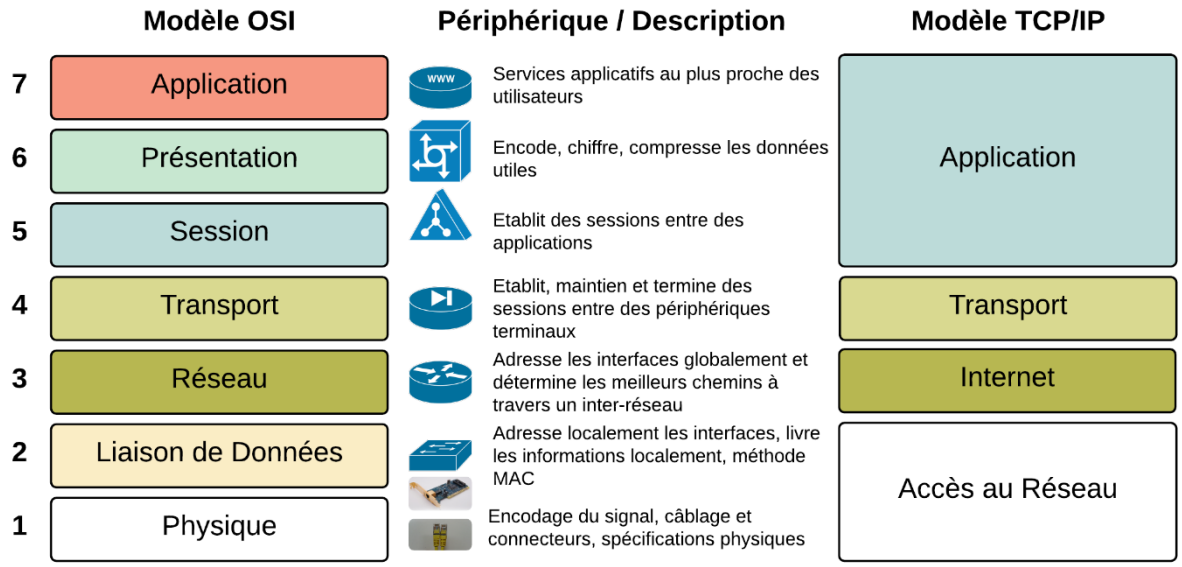
> Les flux doivent être initiés des zones les plus sensibles vers les zones les moins sensibles.

WAF – Web Application Firewall

Si on voit un WAF, c'est probablement que vous n'aurez pas assez bien fait votre boulot de développeurs.

Les pare-feu applicatifs filtrent, inspectent et tracent les flux au niveau des protocoles applicatifs, comme HTTP ou FTP.

C'est-à-dire au niveau 7 du modèle OSI.



Ils combinent des règles de filtrage et des détections par signatures.

Les flux chiffrés ne seront pas analysés sauf…

Question	Comment peut-on faire pour analyser des flux chiffrés en TLS ?
Réponse	Il est possible de faire en sorte de terminer la session TLS au niveau du WAF, ou sur un équipement en amont. Si le flux doit être chiffré jusqu'au bout, on peut ensuite remonter la session TLS vers l'application en *back-end* (cela s'apparente alors à un *Man-in-the-Middle* « légitime »).

Au-delà du filtrage, ils permettent de détecter et bloquer certaines attaques applicatives, comme les SQLi, DoS, directory traversal, XSS, etc.

Le WAF *open source* le plus connu est ModSecurity : https://modsecurity.org

Il existe par exemple des Core Rule Set qui implémentent des détections des vulnérabilités du Top 10 de l'OWASP pour ModSecurity

Autre exemple, le standard PCI DSS (*Payment Card Industry Data Security Standard*) incite aussi à l'usage de WAF.

Ils sont compliqués à configurer: pour éviter les faux positifs, il faut souvent une connaissance fine de l'application qu'ils protègent.

On commence souvent par un mode apprentissage, avant de les passer en mode bloquant.

Exemple de règle ModSecurity pour bloquer les XSS :

> Recherche d'un pattern <script>, blocage et génération d'un message "XSS Attack" avec un code 404
>
> SecRule ARGS|REQUEST_HEADERS "@rx <script>" id:101,msg: 'XSS Attack',severity:ERROR,deny,status:404

À noter qu'au fil des années la frontière entre pare-feu et pare-feu applicatifs n'est plus très nette.

Avant de penser à résoudre les défauts des applications en mettant un WAF devant, regardez s'il n'y a pas des vulnérabilités connues sur vos développements.

Exemple de XSS et comment s'en protéger… avant d'utiliser un WAF :

http://cryptosec.org/docs/test/form.php

IDS / IPS

Une sonde IDS (*Intrusion Detection System*) est un dispositif qui surveille le trafic réseau pour détecter des signes de malveillance ou des violations des politiques de sécurité.

Elle génère des traces (des logs) et des alertes.

Les plus connus sont Snort (https://www.snort.org) et Suricata (https://suricata-ids.org)

Les sondes sont placées à des points clés du réseau. Attention à ne pas en faire des SPOF (*Single Point of Failure*). Une possibilité est par exemple de leur faire analyser le trafic après l'avoir dérivé, ou copié (*port mirroring* sur un switch – commutateur).

À noter également que les flux chiffrés ne seront pas analysés.

Méthodes de détection des IDS / IPS :

- Par signatures, par exemple en cherchant certains motifs, des séquences d'octets ou des séries d'instructions connues comme étant malveillantes. Défaut : ne détecte que les attaques connues / Très sensible au polymorphisme des attaques, comme l'insertion de code mort ou les changements d'encodage ;
- Par détection d'anomalies par rapport à ce qui aura été « appris » comme étant « normal ». Défaut : les faux positifs.

Les sondes qui savent bloquer un flux sont appelées IPS (*Intrusion Prevention System*).

Le blocage peut se faire par reconfiguration d'un pare-feu, ou en bloquant un flux si la sonde est en coupure.

Pour les esquiver, les attaquants pourront essayer de :

- Fragmenter les paquets envoyés ;
- Changer les ports par défaut ;
- *Spoofer* ou *proxyfier* l'adresse source ;
- Modifier les motifs (*patterns*) d'attaque (par exemple modification de la charge utile – le *payload*) (exemple simpliste d'encodage : GET %65%74%63/%70%61%73%73%77%64).

Question	Que signifie GET %65%74%63/%70%61%73%73%77%64 ?
Réponse	En HTTP, il y a plusieurs méthodes : GET, POST, PUT, HEAD, DELETE, PATCH, OPTIONS. Les plus utilisées sont GET et POST. GET demande des données à une ressource, dans l'URL ; POST envoie des données, dans le *body* d'une requête HTTP.

	Et la suite ? C'est une chaine de caractère *URL encoded* : https://www.w3schools.com/tags/ref_urlencode.ASP Avec Burp par exemple GET etc/passwd

Outre les faux positifs (comme les trames malformées du fait de bugs), il y a le problème des alertes non traitées (*missed alarms*) et du bruit qui noie les détections de cas réels.

Remarque importante – déjà évoquée, mais sur laquelle nous insistons – concernant les produits de sécurité :

> La sécurité n'est jamais faite de logiciels, de produits ; la sécurité ce sont des processus.

Il faut mettre en place des processus et une organisation, les logiciels, les produits ne sont que des moyens.

Filtrage des courriels

Les virus et codes malveillants (*malwares*) sont un risque connu depuis longtemps, ils arrivent souvent par courriel.

Le spam est un parasite très gênant dont toute organisation souhaite se prémunir.

Comme la publicité, les spams sont gênants mais ne contiennent en principe pas de vecteur malveillant (mais ils peuvent contenir des tentatives de manipulation participant à des fraudes). À noter que la grande majorité des courriels échangés aujourd'hui appartiennent à l'une de ces deux catégories.

La majorité des attaques ciblées commencent par des courriels envoyés à des victimes contenant des pièces jointes malveillantes (*phishing* ou *spear phishing* – courriels de *phishing* ciblés). Ces pièces jointes peuvent être des documents Office avec macros VBA (Visual Basic for Applications), ou des documents PDF contenant du JavaScript…

Ces pièces jointes peuvent – parfois – être détectées par les passerelles.

Filtrer les courriels entrants est un enjeu important.

En général les messages sont traités en chaine, chaque étape pouvant bloquer le courriel. Par exemple :

- Identification si spam (réputation de l'expéditeur, analyse du message) : on commence par mettre en œuvre ce qui est le moins couteux : recherche d'erreurs protocolaires / puis on passe aux analyses de réputation (List Relay Blocking List – RBL) / puis on passe aux analyses du contenu (pour détecter par exemple qu'une compagnie américaine me parle, mais depuis la Chine, ou l'inverse) ;
- Vérification anti-virus / *antimalware* ;
- Éventuellement *sandboxing* des pièces jointes (une *sandbox* est un environnement maitrisé et cloisonné, souvent une VM, dans lequel on peut lancer des process / logiciels en minimisant les risques. En sécurité, elles sont utilisées pour analyser des *malwares.* Exemple de *sandbox* : https://cuckoosandbox.org).

Si on gère une infrastructure qui envoie des courriels, un des enjeux majeurs est de ne pas se faire *blacklister*. Il s'agit souvent de ne pas être un relais SMTP ouvert sur Internet, ni un quelconque moyen pour des *spammer* d'envoyer des mails à notre place. En cas de *blacklisting*, c'est tout le réseau auquel appartient la machine qui est banni… Très dur d'en sortir.

Deux exemples de solutions antispam et antivirus libres :

- SpamAssassin ;
- ClamAV Antivirus.

Filtrage des accès web

Dans la plupart des organisations existent des proxys web (à l'origine pour résoudre des problèmes de bande passante).

Les proxys web relaient les requêtes des utilisateurs au sein d'un LAN (Local Area Network).

Ils sont devenus des équipements de sécurité essentiels.

Beaucoup d'attaques utilisent les connexions des postes de travail vers Internet, et donc aussi les proxys :

- L'attaque se fait via un site web compromis ou un document téléchargé qui est piégé ;
- Le proxy web peut servir à récupérer la « charge utile » (*payload*) d'un *malware*, récupérée par un *dropper* (composant d'un *malware* en charge de l'installation de la charge utile, qu'il le contienne ou qu'il le télécharge) ;
- Les communications entre le poste de travail et le serveur qui permet de le contrôler (*Command & Control* – C2) transitent via le proxy web ;
- Des logiciels comme TeamViewer ou AnyDesk (télémaintenance, bureau à distance) peuvent être utilisés dans des schémas d'attaque.

Comme dans le cas des passerelles de messagerie, filtrer les accès sortants web est essentiel.

Les méthodes d'analyses peuvent être :

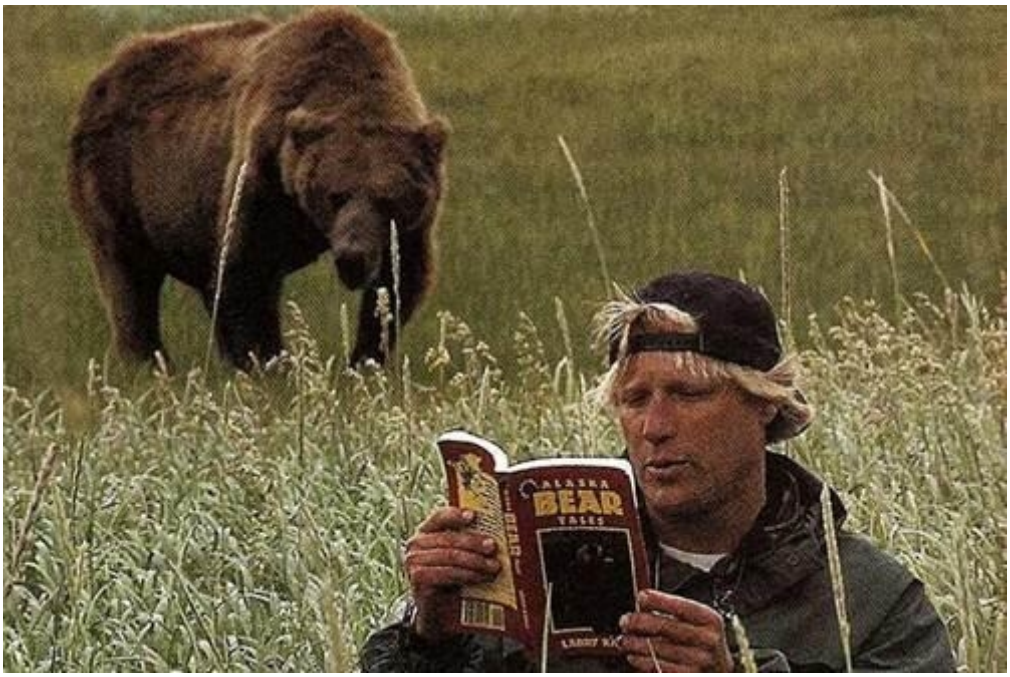
- Analyse des protocoles ;
- Analyse du contenu, en particulier recherche de signatures et motifs malveillants ;
- Blocage de certains protocoles, types MIME, de certaines extensions de fichiers, selon une politique de sécurité ;
- Système de réputation et de catégorisation des sites.

Les fichiers chiffrés ne peuvent être analysés, et les sessions TLS doivent être déchiffrées pour pouvoir être inspectées.

Mais comme pour le filtrage au niveau bas des couches réseau, souvenez-vous d'une banalité :

> Il est impossible d'interdire ce que vous devez autoriser. Et donc, sachez accepter ou faire accepter les risques identifiés.

(Illustration du « Théorème de l’ours »)



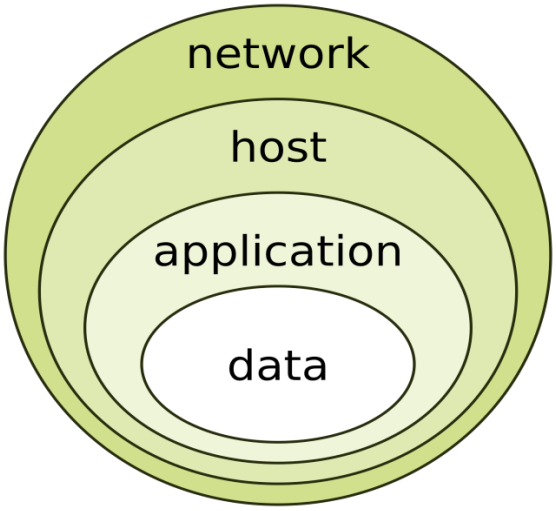
VII. Défense en profondeur

La défense en profondeur est aussi un terme d'origine militaire. L'idée est de mettre des défenses au cœur du système d'information (SI), et pas uniquement en périphérie. Cela vise à ralentir ou stopper un attaquant qui aurait percé les défenses périmétriques.

(Définition d'un SI : ensemble de dispositifs informatiques qui permettent de collecter, stocker, traiter et distribuer de l'information au sein d'une organisation)

On modélise en général la défense en profondeur en quatre activités :

- Sécuriser le réseau ;
- Sécuriser les hôtes (serveurs, postes de travail) ;
- Sécuriser les applications ;
- Sécuriser les données.



Par exemple, une action de défense en profondeur consistera à sécuriser les postes de travail (durcissement, antivirus, HIPS, etc.) au sein d'un SI qui est déjà protégé de l'extérieur. Ou encore le cloisonnement au sein d'un SI, en sus du filtrage vis-à-vis de l'extérieur. L'hypothèse est que les défenses externes peuvent être défaillantes, et qu'il faut que l'attaquant soit confronté à des barrières internes.

En filigrane de toute cette partie, il convient de conserver deux principes à l'esprit :

> En sécurité il est essentiel de prioriser. Ne cherchez pas à atteindre le risque zéro. Concentrez-vous sur les mesures les plus efficaces.

> La sécurité n'est jamais faite de logiciels, de produits ; la sécurité ce sont des processus.

Cloisonnement interne

Un réseau interne, un LAN, peut-être « à plat » ou cloisonné.

S'il est « à plat », toute @IP peut être jointe par toute @IP.

Le cloisonnement réseau consiste à définir des sous-réseaux différents, et à filtrer les flux entre ces sous-réseaux. Des flux peuvent être interdits, ou certains flux seulement être autorisés :

- Le cloisonnement réseau commence en structurant son réseau ;
- Ce n'est que sur la base de cette structure que l'on met en place des règles de filtrage.

Les VLAN (*Virtual Local Area Network*) sont des réseaux virtuels définis sur les *switchs* (commutateurs). Ils permettent de restreindre les communications entre les systèmes selon des règles.

La segmentation peut se faire grâce :

- Aux ports Ethernet des *switchs* (les VLAN sont affectés à des ports sur les *switchs*, les *switchs* sont reliés entre eux par un lien *trunk* pour véhiculer les étiquettes des VLAN) ;
- Aux adresses MAC des systèmes (sachant que ces identifiants peuvent facilement être modifiés).

Les communications entre différents réseaux doivent passer par un routeur.

À noter que les VLAN ne sont pas uniquement définis pour la sécurité : ils permettent aussi de faciliter la gestion des réseaux, d'en améliorer les performances (bande passante, fragmentation)

La norme 802.1q permet de définir des *tags* dans les entêtes des trames Ethernet (des VID – *VLAN Identifier*) et des liens *trunk* qui permettent porter des VLAN sur plusieurs *switchs*.

On peut utiliser des VLAN pour créer des zones de criticité différente, pour héberger des environnements différents (production, intégration, développement…), etc.

Antivirus

Logiciels, parfois appelés *antimalwares*, destinés à détecter, stopper et supprimer les logiciels malveillants (virus, vers, *keyloggers*, *rootkits*, *ransomwares*, chevaux de Troie…).

Exemples d'antivirus : https://www.virustotal.com

Les *malwares* détectés peuvent être bloqués, supprimés ou mis en quarantaine.

La quarantaine est une pratique sanitaire très ancienne.

La récurrence de concepts anciens importés dans le domaine de la cybersécurité mène à une considération importante :

> Ne pensez pas uniquement par déductions et inductions. La pensée par analogie est souvent très fructueuse.

Les méthodes utilisées par les éditeurs d'antivirus (AV) sont souvent obscures et peu ou pas documentées.

Détections basées sur les signatures :

> C’est la méthode historique. Lorsqu’un nouveau virus est identifié, les experts des éditeurs d’AV les examinent, les nomment (W32.Appenex!inf, W32.Duqu.B, Win32 Conficker, Zeus, W32.MyDoom@mm, Trojan.Vundo, Trojan.Vundo.B…) et en créent une signature qui leur permet de les détecter (la signature peut être le condensé d’un fichier ou des portions de son code aisément identifiables).

Mais de plus en plus de *malwares* savent se faire évoluer eux-mêmes, chiffrer des parties de leur contenu…

Méthodes avancées (« heuristiques ») :

> Méthodes permettant de détecter des membres d’une même famille de *malwares* (variantes), par exemple en identifiant des séquences de code communes (mais situées à des endroits différents), en observant leur comportement (à la recherche d’actions suspectes : suppressions de fichiers, lancement de multiples processus…).

Les AV ne détectent pas tous les *malwares*. En particulier, les attaques de type « *0 days* » ne seront pas détectées par la méthode des signatures (une analyse comportementale peut cependant réussir).

De plus, de nombreux attaquants utilisent dans leurs protocoles d’attaque des outils faisant partie de listes d’exceptions des analyses AV (« *whitelistés* » ; cf. le cas Target).

Les AV embarquent parfois des fonctions de détection comportementales – ou « heuristiques » – plus poussées, dites « HIPS » (cf. plus bas).

Question	Tous les dispositifs de détection et de blocage introduisent un risque. Lequel ?
Réponse	Les faux positifs. Pour rappel, de façon générale (c'est-à-dire que ce n'est pas propre aux AV) : - faux positifs : test positif à tort ; - faux négatif : test négatif à tort ; - vrai positif : test positif à raison ; - vrai négatif : test négatif à raison.

Il existe des solutions pour effectuer des tests antivirus non locaux. Par exemple, au niveau d'un proxy web via des requêtes ICAP (*Internet Content Adaptation Protocol*) vers un serveur d'AV.

Si vous voulez tester si un ordinateur ou une infrastructure détecte un virus, plutôt que d'utiliser de vraies souches vous pouvez utiliser des fichiers de test EICAR (constitués par *l'European Institute for Computer Antivirus Research*).

Si vous voulez savoir si un fichier est connu par des AV, le site de référence est : VirusTotal (https://www.virustotal.com). Ce service regroupe plus d'une trentaine d'antivirus.

Attention : si vous soumettez un fichier, les résultats concernant ce fichier seront accessibles aux abonnés, ce qui peut poser des problèmes de confidentialité.

Il est également possible de soumettre des condensés. Le service dispose d'API pour automatiser les soumissions.

Remarque : VT contient également plusieurs dizaines de moteurs de catégorisation d'URL auxquels il est possible de soumettre une URL pour évaluer sa dangerosité.

Les antivirus sont désormais souvent accompagnés de modules d'EDR : Endpoint Détection and Reaction. Ces logiciels mettent en œuvre des stratégies plus larges que les AV pour détecter des menaces, et offrent des fonctions de réaction, de blocage des menaces.

HIPS

Les *Host Intrusion Prevention System* (HIPS) permettent la surveillance des évènements sur un ordinateur, à la recherche de signes d'actions malveillantes.

Similaire à un pare-feu, mais agit sur les changements et les processus (au lieu du réseau).

Les changements, qui peuvent résulter de l'action d'un *malware*, peuvent être :

- Accès en lecture écriture à des fichiers protégés ;
- Accès à la mémoire utilisée par d'autres processus pour injecter du code malveillant ;
- Faire réaliser des actions malveillantes à des programmes légitimes (mail, navigateur…) ;
- Modifications de clés de registres ;
- Arrêt de logiciels de sécurité, comme l'antivirus ;
- Détection de débordement de pile ;
- Connexions suspectes.

L'HIPS doit avoir des droits Administrateur pour pouvoir bloquer l'exécution d'un *malware*.

Il peut aussi embarquer des fonctions de filtrage relevant du pare-feu local.

Mal configuré, un HIPS – comme toute solution de sécurité – peut lourdement handicaper l'expérience utilisateur.

VPM (*Vulnerability and Patch Management*)

Il s'agit du processus de gestion continue des vulnérabilités techniques. C'est un pilier de ce que l'on appelle le maintien en conditions de sécurité (MCS).

Tous les jours sont publiées des vulnérabilités affectant les OS, les logiciels, les *firmware*…

Ces vulnérabilités peuvent en général être corrigées (lorsque c'est possible et faisable), de deux façons :

- Évolution du logiciel, c'est-à-dire application d'un correctif de sécurité ou montée de version (vers une version qui corrige le problème) ;
- Modification du contexte (par exemple de l'architecture) ou de la configuration des systèmes vulnérables.

Les vulnérabilités sont en général publiées selon une certaine nomenclature, les CVE (*Common Vulnerabilty Exposure*).

Exemple de CVE : « ShellShock », un bug dans le shell Bash Unix, permettant à un attaquant de passer des commandes (un utilisateur peut exécuter des commandes qui ne devraient pas lui être accessibles) : https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271

À cette description est associée une estimation de la criticité selon une échelle, le CVSS (*Common Vulnerability Scoring System*)

Exemple CVSS sur « ShellShock » :

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-6271

https://www.cvedetails.com/cve/CVE-2014-6271/

Pour comprendre comment se calcule un score CVSS :

> https://nvd.nist.gov/cvss/v2-calculator?
>
> (Base Score | Impact Subscore | Exploitability Subscore | Temporal Score | Environmental Score | Modified Impact Subscore |Overall CVSS Score)

En général, si ce sont les éditeurs ou fabricants qui publient (*disclose*) ces vulnérabilités, ils publient en même temps un correctif de sécurité (ou *patch*) qui les corrigent.

Parfois, ils font des recommandations, des modifications de la configuration qui permettent de réduire le risque en attendant un correctif.

Une vulnérabilité sur un logiciel?

Il peut s'agir :

- d'un *bug* (ou bogue);
- d'une mauvaise configuration ;
- d'une erreur de conception ;
- d'une vulnérabilité volontairement introduite, une *backdoor* (ou porte dérobée) ;

qu'un chercheur en sécurité ou un attaquant parvient à exploiter à des fins malveillantes (ça peut bien sûr aussi être une erreur de configuration).

Les éditeurs communiquent sur leurs vulnérabilités, mais elles peuvent aussi être révélées par des sites indépendants, des chercheurs, des institutions…

La première étape de tout processus de VPM est de s'assurer que l'on va être informé en cas de publication d'une vulnérabilité sur l'un des logiciels que nous utilisons. Cela peut se faire par un processus de veille manuelle, ou par exemple en inscrivant ses logiciels à une source d'information spécialisée.

Exemples de sites pour recherche manuelle de vulnérabilités :

http://www.securityfocus.com/bid

https://www.exploit-db.com

https://www.shodan.io (pour les IoT)

Le déploiement des correctifs devrait être le plus automatisé possible.

Pour les systèmes critiques, des tests de non-régression peuvent être utiles : les incidents suite au déploiement de correctifs de sécurité sont rares, mais cela arrive. D'autant plus si les éditeurs proposent des mises à jour qui couplent sécurité et fonctionnel. En tout état de cause, déployer un correctif de sécurité est un changement.

Durcissement (*hardening*)

Le durcissement des systèmes consiste à en réduire la surface d'attaque, c'est-à-dire les potentielles vulnérabilités, en modifiant les configurations.

Des actions typiques de durcissement peuvent être :

- Suppression ou désactivation de comptes non nécessaires ;
- Modifications des mots de passe par défaut ;
- Désinstallation ou désactivation de logiciels et modules non nécessaires ;
- Désactivation ou suppression des services non nécessaires ;
- Désactivation des versions non sures de protocoles ou services ;
- Restrictions d'accès à certaines ressources ;
- Restriction de l'usage des droits administrateur ou root ;
- Pas de droits d'exécution là où sont positionnés des droits d'écriture | Pas de droits d'écriture là où sont positionnés des droits d'exécution ;
- etc.

Ces principes de durcissement peuvent s'appliquer aux OS, aux *firmwares*, aux serveurs web, aux applications, aux bases de données, aux objets connectés, aux systèmes industriels, aux *smartphones*…

La gestion de parc permet de forcer de manière régulière une politique de sécurité uniforme.

Un processus de durcissement bien maitrisé peut suivre trois grandes étapes :

- Définition d'une politique, papier uniquement ;
- Déploiement partiel des composants durcis ou des configurations durcies, et observation ;
- Généralisation, et gestion au quotidien des composants durcis (évolutions, dérogations, etc.)

Dans le monde Windows, beaucoup d'actions de durcissement peuvent se faire par des GPO (*Active Directory Group Policy Objects*).

Dans le monde UNIX / Linux, Puppet pourra servir à déployer et redéployer des configurations système (à intervalles réguliers ou manuellement)

Le durcissement consiste aussi à définir des paramètres cryptographiques solides. La cryptographie étant affaire complexe, mieux vaut se référer à des spécialistes pour savoir quelles longueurs de clés et quels algorithmes utiliser : www.keylength.com/fr

Il existe également un projet visant à définir des configurations crypto « prêtes à l'emploi » pour divers logiciels : https://bettercrypto.org/ (en 2020, le site ne semble plus régulièrement maintenu)

NAC

Les solutions de *Network Access Control* (NAC) permettent de contrôler les accès des ressources à un réseau.

Ici encore, il peut y avoir plusieurs types de NAC, selon le niveau de sécurité exigé. Par exemple :

- Absence de DHCP ;
- Contrôle par adresses MAC ;
- Mise en place du protocole 802.1x (avec ou sans certificats).

Il permet de restreindre les accès au réseau, ce qui peut être particulièrement utile lorsque la sécurité physique n'est pas suffisante.

Il peut permettre de conditionner l'accès à un réseau à des contrôles de conformité (présence d'AV à jour, correctifs de sécurité déployés, politiques de sécurité opérationnelles, etc.).

Le NAC peut aussi faciliter certaines opérations de réaction suite à un incident majeur (par exemple en excluant du réseau certains types de machines).

Chiffrement des données (stockées)

Le chiffrement permet de rendre des données inintelligibles.

En simplifiant, il y a trois sortes de chiffrement sur disque :

1- Le chiffrement des volumes de stockage (par exemple tout un disque, une base de données) ;
2- Le chiffrement de containers (par exemple une partition, des tables d'une base de données) ;
3- Le chiffrement de fichiers ou de données unitaires (ou des données avant de les enregistrer en base de données par exemple).

Pour prendre une analogie avec un bureau du début du XXe siècle :

1- La porte du bureau est fermée à clé. On ne peut pas entrer travailler sans la clé ;
2- Le bureau est ouvert, mais les notes papier importantes sont dans un caisson fermé à clé. Pour travailler, il faut l'ouvrir ;
3- Le bureau est ouvert, le caisson aussi. On peut accéder à tout. Mais les notes sensibles sont chiffrées grâce à une machine Enigma.

Chiffrer toute une partition ou tout un disque dur permet de protéger les données en cas de vol de l'ordinateur par exemple, sans se préoccuper si on a chiffré tel ou tel fichier.

La plupart des *frameworks* de développement contiennent des fonctions de chiffrement.

Ne réinventez pas la cryptographie. Jamais.

Intégrée à Windows, on pourra utiliser BitLocker. Cette solution sait interagir avec les cryptoprocesseurs TPM (*Trusted Platform Module*).

On pourra aussi utiliser BestCrypt, qui sait chiffrer les volumes ou des disques virtuels.

À noter que BestCrypt vient accompagné de BCWipe, outil d'effacement sécurisé (par réécriture multiple).

On trouvera également VeraCrypt (https://www.veracrypt.fr), qui est un *fork* de TrueCrypt (arrêté en 2014).

Pour chiffrer ses mails, on pourra utiliser le standard OpenPGP (RFC 4880) dont GnuPG est la première implémentation.

Pour chiffrer ses mots de passe : https://keepass.info. On peut également utiliser des solutions en ligne, comme LastPass (payante) ou Bitwarden (https://bitwarden.com/ ; Open source et dont le service en ligne est partiellement gratuit).

Ou bien le format S/MIME, fonctionnant avec des certificats X.509

(Interlude)

Ayez toujours en tête que :

> La sécurisation doit être homogène : la sécurité globale est toujours celle du maillon le plus faible.

(/interlude)

***Virtual Private Network* (VPN)**

Un VPN est une extension d'un réseau privé à travers des réseaux non sûrs, comme Internet.

Outre la connexion entre plusieurs réseaux, les solutions de VPN permettent à des utilisateurs de se connecter à un SI de façon distante, comme s'ils étaient localement présents.

Le terme peut aussi désigner un tunnel à travers lequel on peut se connecter depuis son accès Internet local pour ressortir ailleurs, en évitant surveillance et censure locale, et avec un certain anonymat. On peut trouver une liste des services (payants) de VPN ici : https://www.vpnspy.net

Les deux protocoles les plus communément utilisés pour mettre en place des VPN sont IPSec et TLS.

Sécurisation de flux : TLS, IPSEC

Transport Layer Security (TLS)

TLS (*Transport Layer Security*) est un protocole de sécurisation au niveau applicatif (couche 6 OSI, couche présentation), successeur de SSL.

C'est le S de HTTPS, typiquement sur le port 443.

Il fonctionne avec des certificats X.509 : au moins un certificat « serveur », mais peut aussi utiliser des certificats « clients ».

Ce protocole assure :

- La confidentialité des données transmises ;
- L'intégrité des données transmises ;
- L'authentification du serveur, éventuellement celle du client.

TLS est défini par deux RFC : RFC 5246 et RFC 6176

Le handshake

Toute session TLS commence par un *handshake* dont les étapes sont les suivantes :

- Le client contacte le serveur et envoie une liste d'algorithmes supportés (*cipher suite*) ;
- Le serveur répond après avoir choisi des algorithmes d'échange de clés, de chiffrement et de condensation ;

- Le serveur envoie également son certificat, qui contient son nom de serveur, sa clé de chiffrement et l'autorité de certification (AC) émettrice ;
- Le client vérifie la validité du certificat ;
- Pour générer les clés de sessions, soit le client chiffre un nombre aléatoire avec la clé du serveur (dont sera dérivée la clé de session), soit il utilise l'algorithme Diffie-Hellman pour s'accorder avec le serveur sur une clé de session ;
- L'échange peut commencer : tout sera chiffré à l'aide de cette clé de session.

Aujourd'hui, TLS est utilisé pour sécuriser les liens client / serveur, mais aussi pour établir des sessions « VPN » d'un client vers une passerelle.

Il y a eu des attaques contre SSL-TLS, comme BEAST, CRIME, POODLE ou encore liées à l'utilisation de RC4.

Utiliser si possible les versions TLS 1.2 ou 1.3.

Une façon d'éviter les attaques du type *Man-in-the-Middle* (MitM) ou les conséquences d'une compromission d'une AC est le *certificate pinning* : une étape est ajoutée à la validation par le navigateur d'un certificat serveur, la vérification du condensé de sa clé publique (ou d'un certificat de la chaîne).

Pour détecter s'il y a un *Man-in-the-Middle* TLS: https://checkmyhttps.net

Concernant les longueurs de clés à utiliser : https://www.keylength.com/fr

Si vous doutez de la configuration TLS à mettre en œuvre : https://bettercrypto.org

Un projet visant à promouvoir la généralisation de TLS sur le web met à disposition une AC qui délivre des certificats serveur gratuitement et de façon automatisée : https://letsencrypt.org

À noter également l'émergence d'une initiative portée par Google, *Certificate Transparency,* qui vise à donner des moyens de détecter des certificats délivrés

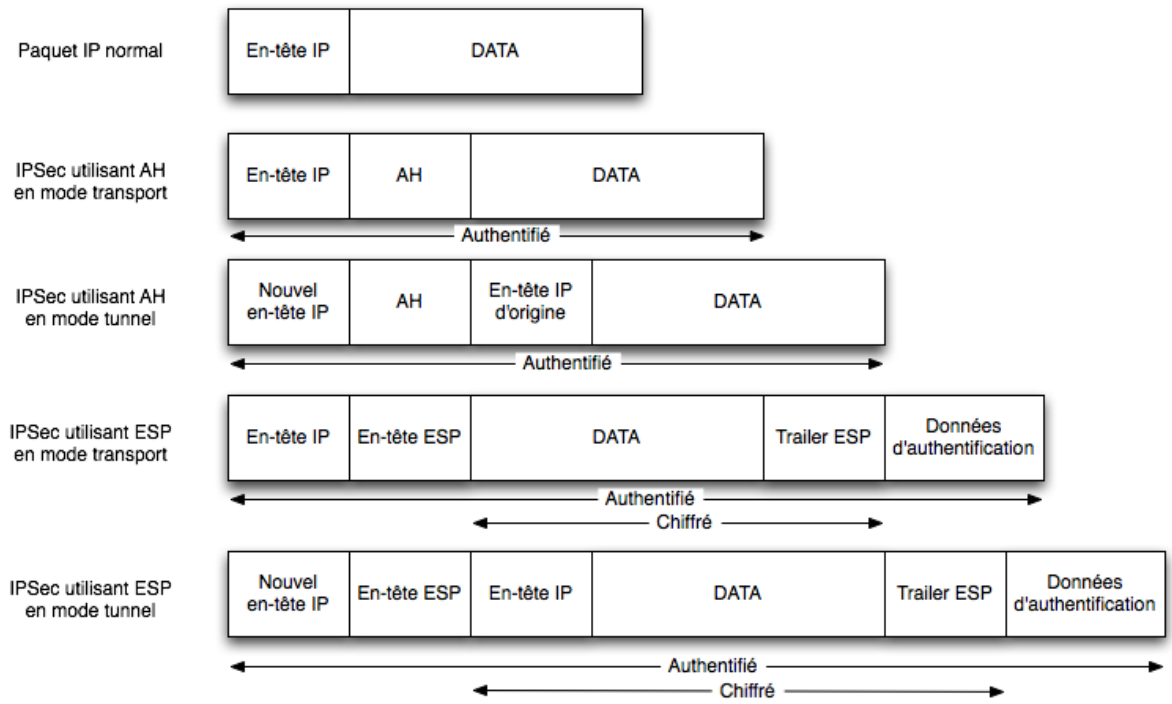
par erreur, ou par des AC compromises par exemple : https://www.certificate-transparency.org

IPSec (Internet Protocol Security)

Série de protocoles qui permet d'authentifier et de chiffrer les paquets d'une communication IP (c'est-à-dire au niveau 3 OSI, couche réseau).

Les principaux protocoles IPSec sont (en résumé) :

- *Authentication Headers* (AH), qui garantit l'intégrité et l'authentification des données ;
- *Encapsulating Security* Payloads (ESP), qui permet de chiffrer les communications, et de garantir l'authenticité des paquets ;
- *Security Associations* (SA), algorithmes et échanges permettant de mettre en œuvre AH et ESP ;
- *Internet Security Association and Key Management Protocol* (ISAKMP), qui permet l'authentification et l'échange de clés peut fonctionner en échange de clés ou par certificats (IKE).



(source : http://igm.univ-mlv.fr/~dr/XPOSE2007/cchamp01_VPN/IPSec.html)

IPSec, défini par de nombreuses RFC, fait partie intégrante de la spécification d’IPv6.

Vu qu’il est « bas » au niveau protocolaire, il permet d’établir des tunnels qui véhiculent tout type de protocoles reposant sur IP.

VIII. Retour sur les vulnérabilités

En général les vulnérabilités résultent :

- d'un *bug* (ou bogue);
- d'une mauvaise configuration ;
- d'une erreur de conception ;
- d'une vulnérabilité volontairement introduite, une *backdoor* (ou porte dérobée) ;

qu'un chercheur en sécurité ou un attaquant parvient à exploiter à des fins malveillantes.

Ci-après, quelques vulnérabilités typiques.

Injections

Une faille d'injection, telle l'injection SQL, OS et LDAP, se produit quand une donnée non fiable est envoyée à un interpréteur en tant qu'élément d'une commande ou d'une requête. Les données hostiles de l'attaquant peuvent duper l'interpréteur afin de l'amener à exécuter des commandes fortuites ou accéder à des données non autorisées.

Une injection SQL, ou SQLi, est l'exploitation d'une vulnérabilité d'une application interagissant avec une base de données. Elle permet d'injecter dans la requête SQL une requête non prévue par le système et permettant d'accéder à des informations en outrepassant des mesures de sécurité.

Question	Exemple d'une injections SQL : création d'utilisateurs, modifications de contenu, scan de port… Pour tester : ‘ Que peut-il se passer ?

Réponse	‘ or 1=1 -- On a fermé une commande et on en a lancé une autre…
	Script d'authentification vis-à-vis d'une base de données:
	# Define POST variables
	uname = request.POST['username']
	passwd = request.POST['password']
	# SQL query vulnerable to SQLi
	sql = “SELECT id FROM users WHERE username=’” + uname +

	“’ AND password=’” + passwd + “’”
	# Execute the SQL statement
	database.execute(sql)
	Si on remplace “uname” par password’ OR 1=1 --, on a
	SELECT id FROM users WHERE username=’username’ AND password=’password’ OR 1=1’
	Et le résultat est transmis à l’application: on a contourné l’authentification.

Cross-site scripting (XSS)

Les failles XSS se produisent chaque fois qu’une application accepte des données non fiables et les envoie à un navigateur web sans validation appropriée. XSS permet à des attaquants d’exécuter du script dans le navigateur de la victime afin de détourner des sessions utilisateur, défigurer des sites web, ou rediriger l’utilisateur vers des sites malveillants

C’est aussi une injection. Les XSS stockés permettent d’injecter du Javascript dans une base de données (ou équivalent), de façon à ce qu’il soit ensuite incorporé et exécuté dans les navigateurs visitant une page qui y fait appel.

https://www.hacksplaining.com/exercises/xss-stored

XSS Réfléchi

Dans ce cas, le code Javascript est passé en paramètre d’une URL. Cela permet de faire exécuter le JS dans le navigateur de la victime sans avoir à écrire quoi que ce soit dans le site. Le site n’est alors que le vecteur, le véhicule, de l’attaque.

Ce vecteur pourra par exemple être envoyé à la victime par courriel (et n’oubliez pas qu’un lien peut pointer vers autre chose que ce qui parait : https://www.lemonde.fr

Démonstration sur les XSS réfléchis :

https://www.hacksplaining.com/exercises/xss-reflected

Références directes non sécurisées à un objet

Une référence directe à un objet se produit quand un développeur expose une référence à un objet d'exécution interne, tel un fichier, un dossier, un enregistrement de base de données ou une clé de base de données. Sans un contrôle d'accès ou autre protection, les attaquants peuvent manipuler ces références pour accéder à des données non autorisées.

Exemple typique, un *path traversal* (traversée de répertoire)

http :// dvwa . zarb . org / vulnerabilities / fi ? page =../../../../.../../../../etc/passwd

Exécution de commande

Parfois, du fait d'une vulnérabilité dans une application, il est possible de faire exécuter des commandes (shell par exemple) à l'ordinateur distant, via des requêtes HTTP.

https://www.hacksplaining.com/exercises/command-execution

Exemple : https://hackingiscool.pl/cmdhijack-command-argument-confusion-with-path-traversal-in-cmd-exe/

Téléversement et exécution de fichiers malveillants

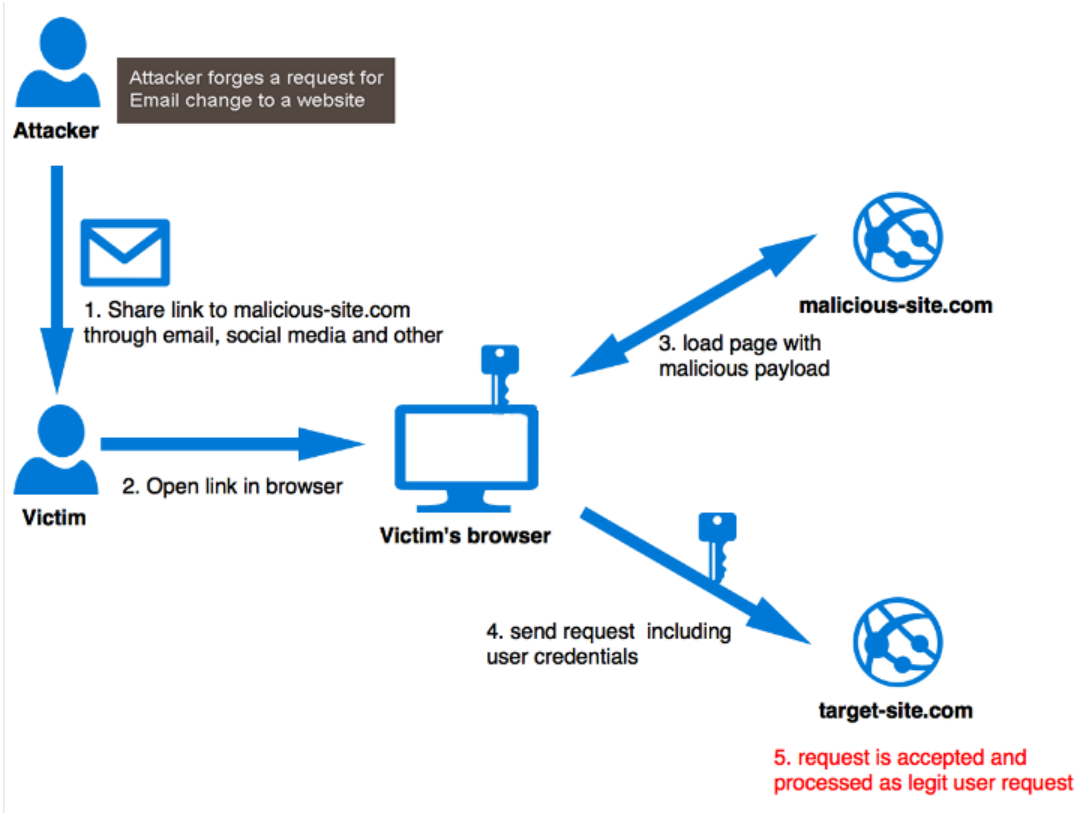
Un code vulnérable à l'inclusion de fichier à distance (RFI - Remote File Inclusion) permet à des attaquants d'inclure du code et des données hostiles, ayant pour résultat des attaques dévastatrices, telle la compromission totale d'un serveur. Les attaques par exécution de fichier malveillant affectent PHP, XML et toute structure qui accepte des noms de fichiers ou des fichiers des utilisateurs.

C'est exactement ce qu'ont fait les attaquants de Target, présentée en début de cours (*upload* d'un webshell)

https://www.hacksplaining.com/exercises/file-upload

CSRF

Une attaque CSRF (Cross Site Request Forgery) force le navigateur d'une victime authentifiée à envoyer une requête HTTP forgée, comprenant le cookie de session de la victime ainsi que toute autre information automatiquement inclue, à une application web vulnérable. Ceci permet à l'attaquant de forcer le navigateur de la victime à générer des requêtes dont l'application vulnérable pense qu'elles émanent légitimement de la victime.



Exemple d'utilisation malveillante: amener un administrateur à changer son mot de passe à son insu. Il faut d'abord identifier l'URL permettant de réaliser l'action, puis l'utiliser pour fournir un lien piégé à la victime.

https://www.hacksplaining.com/exercises/csrf#

Mesures à mettre en œuvre :
https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

Buffer overflow

Si un programme écrit dans un tampon (*buffer*) et qu'il dépasse par erreur la taille attribuée à ce tampon, alors il est envisageable de faire exécuter du code arbitraire. En C, ce peut être par exemple via gets (lecture d'une suite de caractères) ou strcpy (copie d'une chaîne de caractères) utilisés sans validation de la taille des entrées. Outre les protections dans le code, il peut y avoir des protections « système » comme DEP (Data Execution Prevention) et ASLR (Address Space Layout Randomization) (dont le but est essentiellement de rendre le cout d'une exploitation plus important)

https://www.hacksplaining.com/exercises/buffer-overflows

Entrainement

Exercices en ligne:

https://tryhackme.com/

https://www.root-me.org/

https://www.newbiecontest.org/

DVWA : http://www.dvwa.co.uk/

WebGoat : https://owasp.org/www-project-webgoat/

Gruyere : https://google-gruyere.appspot.com/

IX. Politique de sécurité

Une politique de sécurité est un document décrivant les obligations en matière de sécurité.

Un principe général est qu'il faut écrire ce que l'on fait, et faire ce que l'on écrit.

Elle peut définir :

- Les principes de sécurité ;
- Des règles précises de sécurité (par exemple la politique de mots de passe ou les obligations en matière de *Vulnerability and Patch Management*) ;
- Les méthodes et critères de classification de l'information. Pour tous les projets, cette étape est importante : classifiez vos données, par exemple selon des critères comme : public, interne, retreint, confidentiel. Identifiez les données et processus sensibles ;
- Les obligations en matière de respect de la Loi Informatique et Libertés.

Classifiez vos données (publique, interne, retreinte, confidentielle). Identifiez les données et processus sensibles.

Souvent les politiques de sécurité sont trop performatives : du simple fait d'énoncer des exigences, les services sécurité imaginent qu'elles deviennent réalité…

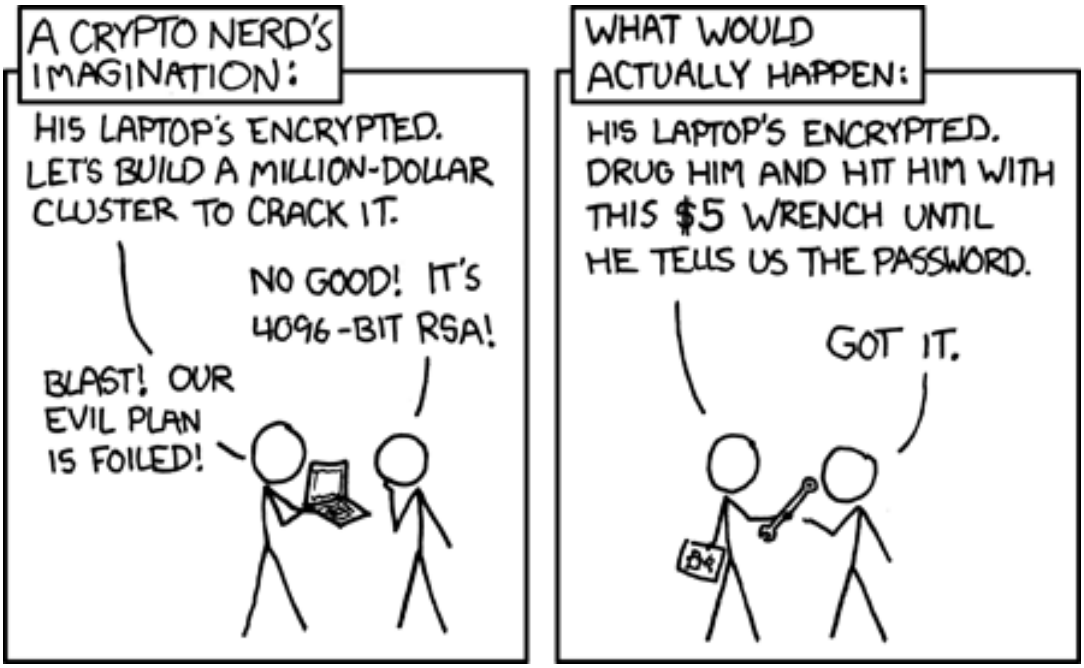
L’objectif d’un corpus documentaire devrait toujours d’être compréhensible, cohérent, utilisable et utilisé.

Dans tous les cas, avant de commencer à rédiger des spécifications ou de développer, il faut lire, essayer de comprendre et prendre en compte les exigences des politiques de sécurité.

Pour définir une politique de sécurité, définir des priorités et des choix stratégiques, préférer les référentiels concrets et construits sur des constats techniques et factuels comme les CIS Controls :

> https://www.cisecurity.org/controls/

plutôt que les « bonnes pratiques » de l’ISO, qui sont souvent trop éloignées de la réalité des attaques.



Pour ne pas se perdre en mesures de sécurité théoriques, n’oubliez pas :

> La connaissance des attaques oriente la défense.

X. Sensibilisation à la sécurité

Certains disent que la seule façon d'atteindre un niveau de sécurité parfait… c'est sans utilisateurs. Mais il y en a.

Les utilisateurs, par leurs erreurs ou parce ce qu'ils sont vulnérables aux tentatives d'ingénierie sociale, sont souvent une cible de choix pour les attaquants.

Aujourd'hui, l'immense majorité des attaques réussies commencent par un courriel envoyé à quelqu'un qu'on essaie de convaincre de cliquer sur un lien, ou d'ouvrir une pièce jointe. Pas par une tentative d'exploitation d'une *0 day* sur un pare-feu.

Il faut donc toujours sensibiliser les utilisateurs au respect d'une certaine hygiène de la sécurité informatique, et s'y astreindre soi-même.

S'il y a un message universel à retenir en matière de sensibilisation des utilisateurs, ce serait : « Réfléchissez avant de cliquer ».

Ajoutons qu'un argument facilite l'appropriation par les utilisateurs des mesures de sécurité : appliquer les mesures de sécurité participe à protéger les biens et processus de l'organisation, de l'entreprise dans laquelle on évolue. Mais cela protège aussi celui qui les applique (par exemple : suite à une attaque ayant impliqué des comptes privilégiés, ceux qui n'en disposaient pas – si la politique de sécurité l'interdisait – seront d'emblée hors de cause).

Un dernier point : les actions pédagogiques pour favoriser le respect des mesures de sécurité seront d'autant plus efficaces que les gens pourront en appliquer les leçons dans leur vie privée.

Enfin, en ce qui vous concerne :

Formez-vous. Demandez des budgets formation, faites valoir vos droits à la formation.
Partagez avec vos pairs, assistez à des conférences sécurité, à l'étranger de préférence. Faites-vous payer le voyage, l'hôtel. Partagez. Pas avec n'importe qui, dans des cercles de confiance et en faisant attention à la confidentialité. Mais partagez.

XI. Sécurité des accès

Les principes généraux d'une bonne sécurité des accès :

- Une identification, authentification, autorisation et traçabilité solide ;
- Moindre privilège : donner les privilèges minimums ;

- Besoin d'en connaitre : ne donner les accès qu'aux personnes en ayant besoin, pour le temps nécessaire, et les retirer ensuite ;
- Séparation des rôles : séparation des taches critiques entre divers intervenants, afin d'éviter l'accumulation des privilèges ; par exemple en implémentant des procédures « 4 yeux ».

Moindre privilège : par principe, donner les privilèges minimums.

Besoin d'en connaitre : ne donner les accès qu'aux personnes en ayant besoin, pour le temps nécessaire, et les retirer ensuite.

Séparation des rôles : séparation des taches critiques entre divers intervenants, afin d'éviter l'accumulation des privilèges ; par exemple en implémentant des procédures « 4 yeux ».

La première étape de la sécurité des accès, l'authentification à des ressources informatiques repose sur quatre étapes, distinctes :

- L'**identification** : il s'agit pour le système d'identifier de façon unique l'utilisateur, typiquement, le nom d'utilisateur (ou *username*) ;
- L'**authentification** : le système vérifie que l'utilisateur est bien celui qu'il prétend être (par exemple parce qu'il exige un mot de passe ou un code PIN) ;
- L'**autorisation** : ce sont les droits positionnés sur le système qui donnent accès aux ressources selon l'utilisateur, typiquement les permissions sur un système de fichier ;
- La **traçabilité** : le système doit conserver la trace des accès afin de savoir qui s'est authentifié, quand, qui a tenté, etc.

Par exemple, lors de l'accès à une application : L'application vérifie qu'elle connait mon nom d'utilisateur, un mot de passe va authentifier que je suis bien celui que je prétends être, et enfin, une fois ces étapes réussies, l'application va m'autoriser à accéder à certaines ressources et à utiliser certains services, sur la base de mon identifiant.

Le mécanisme d'authentification peut reposer sur trois éléments :

- Ce que je sais (un mot de passe par exemple) ;
- Ce que je possède (une carte à puce ou un OTP – *One-Time Password* – par exemple) ;
- Ce que je suis (empreintes digitales ou rétinienne par exemple).

On parle « d'authentification forte » (2FA – *Two Factor Authentication ou MFA – Multi Factor Authentication*) lorsqu'un mécanisme d'authentification met en œuvre deux de ces trois facteurs.

L'authentification forte complexifie le vol d'identifiants, par exemple dérobés via des attaques de type *spear phishing*. Elle ne protège pas uniquement l'application qui la met en œuvre ; c'est tout le SI qui l'héberge qui en bénéficie, parce qu'elle diminue la probabilité que l'application – une fois compromise – devienne un point d'entrée pour l'attaquant.

Quelques modèles de contrôle d'accès :

- **Discrétionnaire** (*Discretionary Access Control*) : basé sur l'identité ou l'appartenance à des groupes, et sur la capacité du possesseur d'un droit à le transmettre à un autre utilisateur ; décentralisé.
- **Obligatoire** (*Mandatory Access Control*): les contrôles d'accès dans ce cas sont imposés par une politique de sécurité, et ne sont pas laissés à la liberté du propriétaire des ressources ;
- **À base de rôles** (*Role-Based Access Control – RBAC*) : dans ce cas c'est l'appartenance à un rôle qui détermine les droits d'accès. Les rôles peuvent refléter la structure d'une organisation ;
- **Basé sur les attributs** (*Attribute-Based Access Control – ABAC*) : dans ce modèle des règles combinant des attributs définissent les droits. Il est possible d'utiliser des opérateurs booléens (if, then) concernant l'initiateur de la requête, la ressource ou l'action.

Question	Pourquoi « chmod 777 » sur le répertoire d'un serveur web est-il dangereux ?
Réponse	Parce que les droits rwxrwxrwx permettraient à un attaquant réussissant à exploiter des vulnérabilités sur le serveur web (par exemple *upload* d'un *webshell et* d'un exécutable) d'écrire et de faire exécuter par le système sur les répertoires accessibles au serveur web. Pour être exact, ce n'est pas le 777 qui est dangereux en soi, c'est la violation du principe de séparation d'écriture et d'exécution.

Question	Quel est le problème avec ces messages en cas d'erreur d'authentification ? - Login for User cryptosec: invalid password - Login failed, invalid user ID - Login failed; account disabled - Login failed; this user is not active
Réponse	Il ne faut pas donner d'informations aux attaquants qui leur permettraient de réduire leurs efforts.

En cas d'échec d'authentification :

- Soit on peut bloquer le compte après un certain nombre de tentatives ;
- Soit on peut introduire une temporisation.

Le blocage a l'avantage de couper court à toute tentative par force brute mais il peut permettre un déni de service en bloquant un grand nombre de comptes.

Accès « dormants » : il est important de prévoir une fonction permettant à un utilisateur d'effacer son compte, ou au système de purger de temps en temps les comptes jamais utilisés.

Les mots de passe ne sont pas la seule façon de s'authentifier.

Exemple : les clés SSH. Un jeu de bi-clés (sous la forme de fichiers) sert à s'authentifier auprès du serveur et à négocier une clé symétrique de chiffrement (à l'aide de l'algorithme Diffie-Hellman).

Question	Quel est l'avantage de l'authentification via des clés SSH par rapport à un classique mot de passe ?
Réponse	Aucun mot de passe ne transite via le réseau. L'accès au fichier de clés se fait en saisissant un mot de passe qui reste local à la machine cliente.

(Illustration de la dialectique Front-end / Back-end)

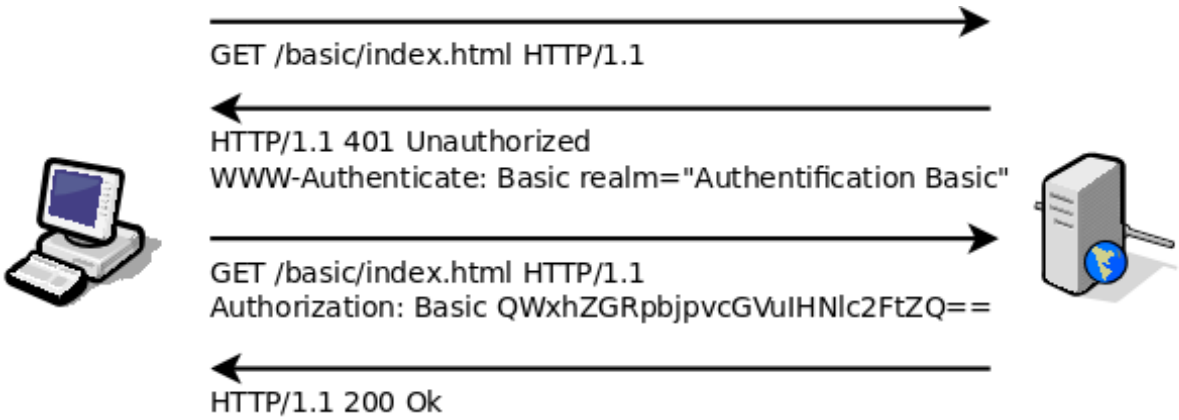
Méthodes d'authentification cliente

Authentification HTTP Basic

Lorsque le client demande à accéder à une page, le serveur demande une authentification au client (HTTP 401), avec un realm=www.le-serveur.org

Le client renvoie un GET avec le nom d'utilisateur et le mot de passe encodé en base64.

Le serveur vérifie le couple identifiant / mot de passe par rapport à une base de comptes quelconques (propre au serveur web, celle du système, celle d'un référentiel distant, etc.).



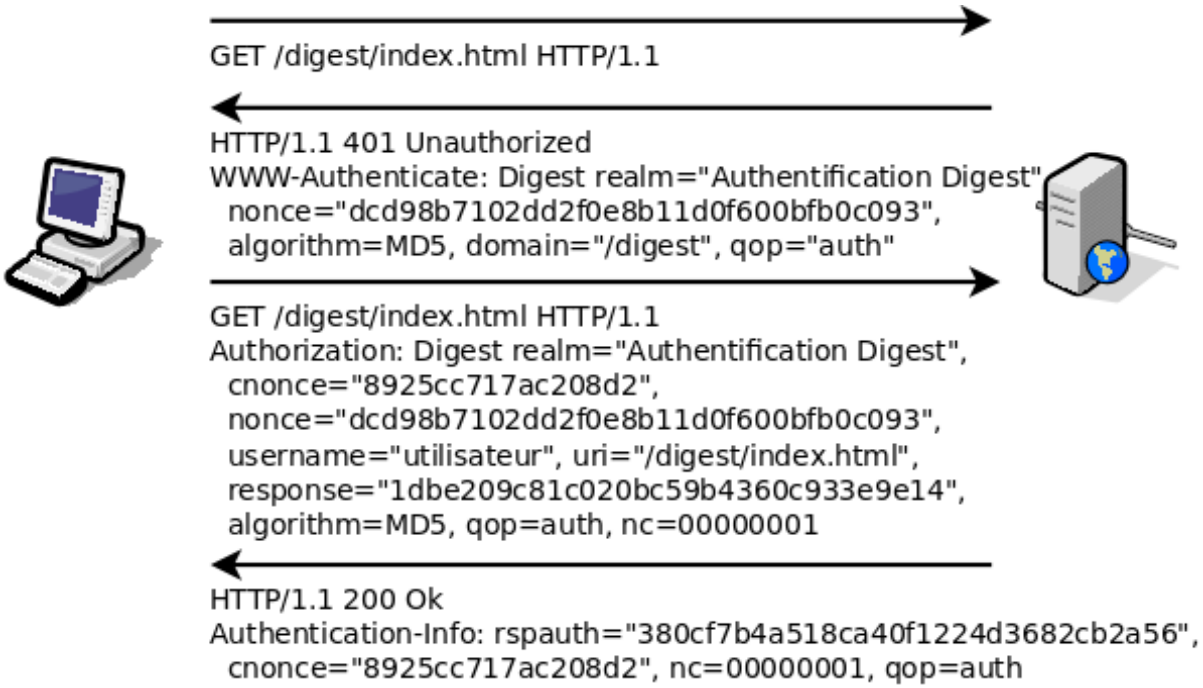
Mécanisme assez faible :

- Authentification en clair ;
- Peut être rejouée ;
- Très faible authentification du serveur ;
- Pas de déconnexion (sauf fermeture du navigateur).

Question	Qu’est-ce qui peut améliorer un peu la sécurité de l'a*uthentification HTTP Basic*?
Réponse	Passer en session chiffrée / intègre, TLS.

Authentification HTTP Digest

Évolution de l’authentification Basic, en mode challenge / réponse : le serveur envoie un « aléa », que le client adjoint au mot de passe avant de le condenser (MD5).



Cette fois, le mot de passe n’est pas envoyé.

Et le rejeu est difficile.

Mais vulnérable à l’attaque de « l’homme du milieu » (*Man-in-the-Middle Attack*), en plaçant un serveur intermédiaire qui intercepte l’aléa du serveur légitime.

Authentification Microsoft intégrée (IWA)

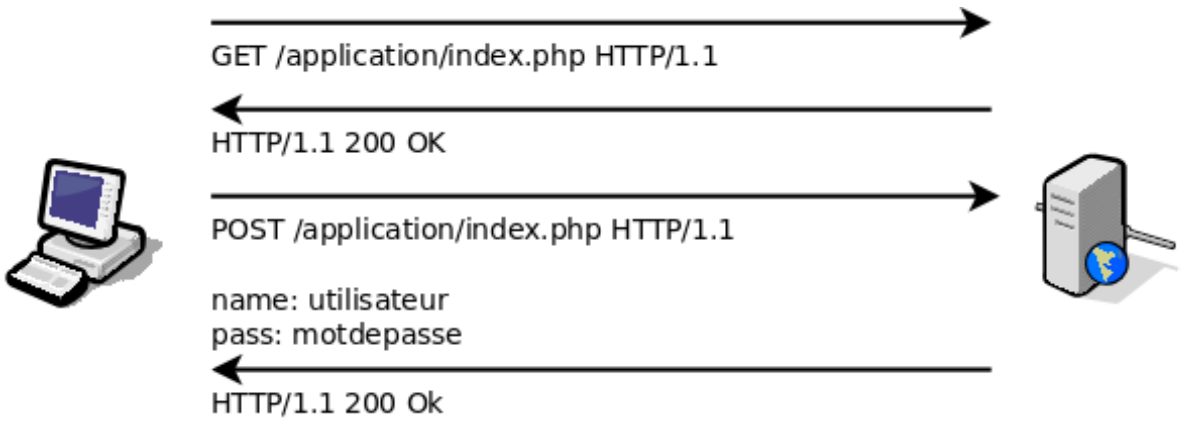
Repose sur le mécanisme d'authentification Windows, protocole challenge / réponse.

C'est du Kerberos ou du NTLM via HTTP ou HTTPS, et on trouve ce moyen d'authentification typiquement dans les intranets Microsoft.

1- L'utilisateur s'authentifie à une machine cliente ;
2- Windows vérifie l'authentification auprès d'un contrôleur de domaine ;
3- L'utilisateur veut aller sur un site ;
4- Le serveur web exige une authentification ;
5- Le client envoie le token d'authentification.

Authentification applicative

Formulaire présenté à l'utilisateur, l'application vérifie le nom d'utilisateur / mot de passe vis-à-vis d'une base de comptes (LDAP, base de données, etc.).

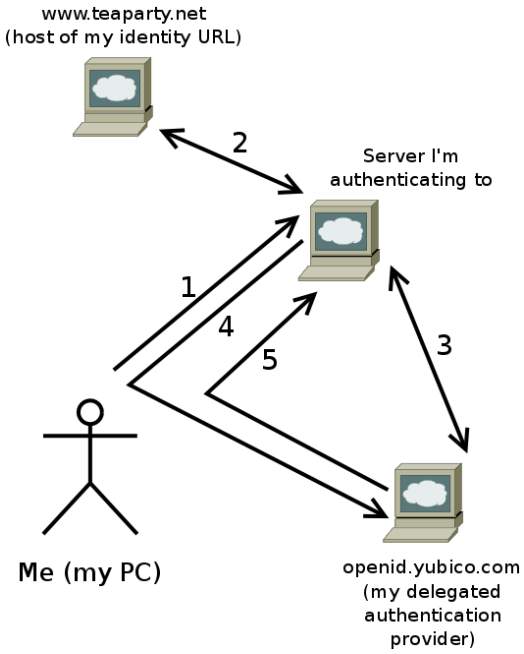


Il faut véhiculer le flux en HTTPS.

La sécurité dépend de la conception et de développements spécifiques, on peut donc plus facilement y trouver des vulnérabilités de type injections (XSS, SQLi…).

OpenID

Un exemple de mécanisme d'authentification décentralisé, qui permet aux développeurs de ne pas redévelopper leur système d'authentification et aux utilisateurs de s'authentifier à de multiples sites sans se réauthentifier à chaque fois.



(https://lwn.net/SubscriberLink/708151/a26c64b1d7ffec65/)

1. L'utilisateur va sur le site sur lequel il veut s'authentifier et déclare qu'il veut s'authentifier en tant que : l'identifiant décrit sur https://www.teaparty.net/ (rien à voir avec les réactionnaires étasuniens) ;
2. Le site récupère des tags dans la page https://www.teaparty.net/ qui lui donnent son identité et son fournisseur d'identité ;
3. Le site contacte openid.yubico.com et POST une requête pour établir une clé secrète partagée (algorithme Diffie-Hellman) ;
4. Le site renvoie le navigateur de l'utilisateur vers openid.yubico.com pour s'authentifier (selon la méthode choisie pendant l'enregistrement) et en ajoutant un aléa qui lui est propre ;
5. Comme l'utilisateur s'est authentifié, openid.yubico.com renvoie le navigateur vers le site avec un token, authentifié par le fournisseur d'identité et spécifique à cette authentification (grâce à l'échange 3.) et à l'aléa.

OAuth

OAuth2 est un protocole – ou *framework* – d'autorisation.

OAuth est souvent utilisé pour que l'on puisse donner à des sites l'accès à des données traitées par d'autres sites, mais sans communiquer son mot de passe sur ces autres sites.

Les utilisateurs peuvent donner au site consommateur l'accès à des données d'authentification provenant d'un site fournisseur, sans révéler logins ou mots de passe.

OAuth date de 2007 et dérive de OpenID. Depuis 2012, OAuth 2.0 est défini par les RFC 6749 et RFC 6750.

Ce protocole est par exemple utilisé par Amazon, Google, Facebook, Microsoft ou Twitter.

Le protocole de délégation d'autorisation permet à un site web, un logiciel ou une application « consommateur » d'utiliser une API d'un autre site web, « fournisseur », pour le compte d'un utilisateur.

Il permet ainsi à un *authorization server* de délivrer des tokens d'accès avec l'approbation de l'utilisateur, le *resource owner*, le tout en HTTP.

A noter que OAuth ne spécifie pas la méthode d'authentification à utiliser.

Fonctionnement

L'user va sur l'application. En général, l'user détient les droits sur le *ressource server*. Il veut utiliser ces ressources, à travers l'application. Mais l'application doit avoir des privilèges sur le *ressource server* pour les récupérer et les lui présenter.
Par exemple, il peut s'agir d'un calendrier (application) qui veut accéder à une boite aux lettres (ressource).

Il y a une façon *dirty* de faire: on donne login / mot de passe à l'application, qui l'utilise pour accéder à la ressource.
Mais cette façon de faire pose plusieurs problèmes :

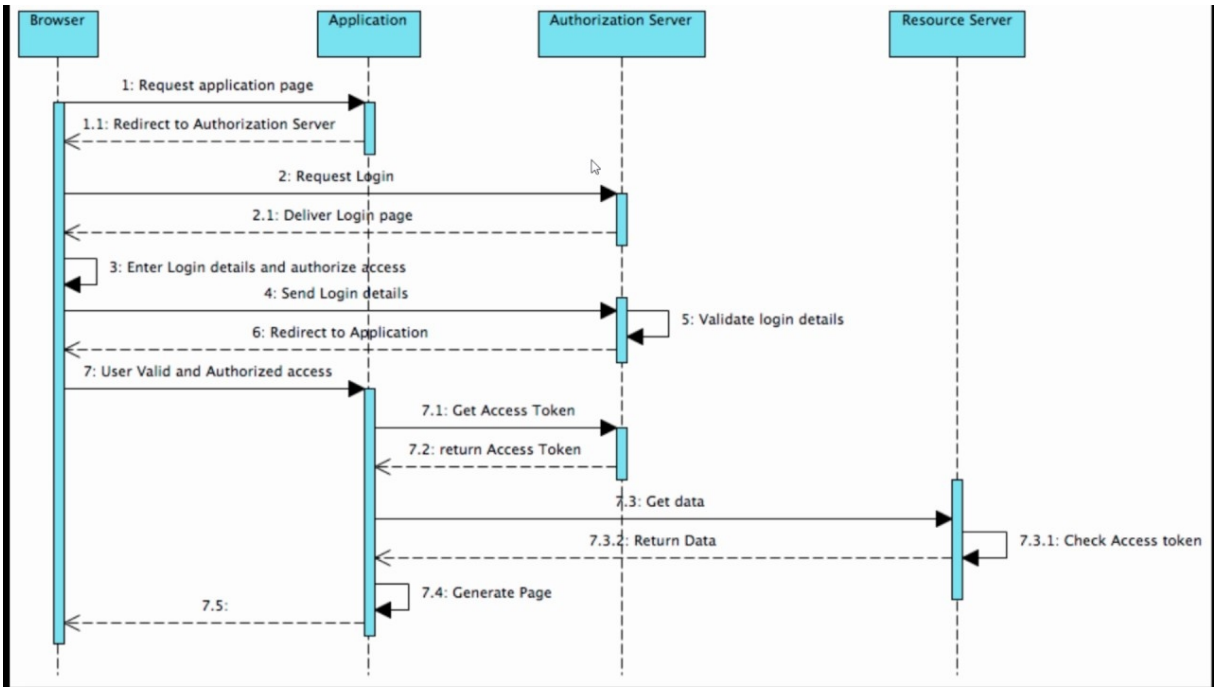
- Pour commencer, on communique à un tiers son mot de passe, ce qui est une mauvaise pratique ;
- Et comment fera-t-on si on veut révoquer ce droit que l'on a donné à l'application ?

- Et que se passe-t-il si l'application fait quelque chose d'interdit, en utilisant le mot de passe ?

Le problème à résoudre est équivalent au suivant :
Vous allez à une soirée, et vous avez oublié votre smartphone. Vous devez consulter de toute urgence quelque chose sur votre webmail. Il y a un ordinateur. Vous demandez à l'hôte de la soirée, que vous ne connaissez pas bien. Il veut bien, mais il tient à ce que ce soit lui qui manipule son ordinateur et saisisse le mot de passe. Vous devez donc lui donner votre mot de passe. Vous aimeriez qu'il existe un moyen de ne lui donner qu'un mot de passe temporaire, qui ne donne accès qu'à l'information dont vous avez besoin, et pas à tous vos courriels ; ou un autre moyen qui vous permette d'obtenir l'information que vous cherchez dans votre webmail, sans lui donner votre mot de passe… C'est ce que fait OAuth.

Cinématique

Les quatre acteurs du protocole sont :
- *resource owner*: utilisateur (navigateur), accorde l'autorisation
- *client* : application, serveur web
- *authorization server*: authentifie le *ressource owner* et délivre des jetons, *access token*
- *ressource server*: serveur ou sont stockées les ressources protégées



(source : https://www.youtube.com/watch?v=L1PDqJkedZ0)

- 1.1 : Rediriger le navigateur vers l'*authorization server* (dans le contexte HTTP)
- 2 : L'utilisateur va voir la page de l'*authorization server*, lui demandant de vérifier ses login / mot de passe

- 6 : Après vérification, *l'authorization server* renvoie vers l'*application* et ajoute un code d'autorisation
- 7.1 : L'*application* émet un POST vers *l'authorization server*, avec ce code
- *7.2 : L'authorization server* renvoie un token d'accès à l'*application.* C'est souvent à cette étape que *l'authorization server* demande à l'utilisateur s'il donne son autorisation
- 7.3 : L'*application* envoie sa requête et le token (dans le header) au *ressource server*
- 7.3.1 : le *ressource server* vérifie le token. Il peut le faire parce qu'il partage une information avec l'*authorization server*.
- 7.3.2 : Le *ressource server* vérifie le token et renvoie les données à l'*application*, qui, finalement les sert au navigateur

Illustration :
https://www.oauth.com/oauth2-servers/server-side-apps/example-flow/

Authentification centralisée et SSO

L'authentification centralisée désigne un mécanisme capable de gérer la fonction d'authentification de plusieurs composants (systèmes, applications, etc.). Typiquement, un annuaire LDAP permet d'assurer cette fonction.

Les avantages de l'authentification centralisée sont :

- Facilité pour les utilisateurs, qui n'ont pas à retenir ou stocker de nombreux noms d'utilisateurs / mots de passe (stockés par exemple dans un fichier chiffré de type KeePass) ;
- Gains de temps et de couts (temps des utilisateurs – saisie des mots de passe et couts des appels au support pour gérer les mots de passe) ;
- Réduction du risque de compromission des mots de passe (chez l'une des multiples entités).

Les risques d'un système d'authentification faible, si les mécanismes d'autorisation adéquats ne sont pas en place, c'est un accès démultiplié au SI en cas de compromission d'un compte. Typiquement, un utilisateur révélant son mot de passe suite à une attaque par *phishing* peut permettre à un attaquant l'accès au VPN, puis aux données utilisateur, etc…

Une évolution de l'authentification centralisée, c'est le *Single Sign-On (SSO)*: non seulement l'utilisateur n'a plus qu'un mot de passe unique, mais en plus il n'a besoin de le saisir qu'une seule fois sur un système pour accéder ensuite à de multiples plateformes et applications.

Un exemple de standard de SSO sur le web est SAML (*Security assertion markup language*), qui permet de naviguer sur plusieurs sites différents en ne s'étant authentifié qu'une seule fois :
https://en.wikipedia.org/wiki/Security_Assertion_Markup_Language

Question	Pourquoi ne peut-on pas simplement utiliser les cookies pour faire du SSO entre des sites différents ?
Réponse	Les cookies permettent aux développeurs de stocker des informations sur l'utilisateur. Exemple d'un *header* HTTP: Set-Cookie: <name>=<value>[; <Max-Age>=<age>][; expires=<date>][; domain=<domain_name>][; path=<some_path>][; secure][; HttpOnly]. À cause de la *Same Origin Policy*, qui n'autorise pas un script d'un site A d'accéder aux cookies d'un site B s'ils ne partagent pas la même « origin », définie par le triplet URI, nom d'hôte, numéro de port.

Kerberos

Protocole de SSO reposant sur une authentification initiale auprès d'un AS (Authentication Server – qui possède les mots de passe des utilisateurs) qui distribue des Ticket-Granting Ticket (TGT).

Ensuite, le service de Ticket-Granting Service délivre des jetons qui pourront être déchiffrés et authentifiés par les services que l'on voulait atteindre.

Pour donner une analogie :

- Un État délivre des passeports à ses citoyens (c'est l'AS qui donne des TGT) ;
- Pour aller dans certains pays, on demande un visa. Le consulat qui délivre les visas vérifie l'authenticité du passeport présenté, et délivre le visa - valable uniquement pour un pays, et pour une durée limitée (ce sont les TGS).

Initialement développé dans le monde UNIX, Kerberos a été adopté par Microsoft et est aujourd'hui le cœur du système d'authentification de cet éditeur (avec quelques modifications par rapport au protocole initial).

Le protocole est défini par la RFC 4120 et quelques autres.

SSO par cartes à puces / tokens crypto :

> Dans ce cas, le secret est simplement stocké dans la puce, et après avoir entré le PIN il peut être utilisé pour s'authentifier auprès de divers services / sites.

Mots de passe

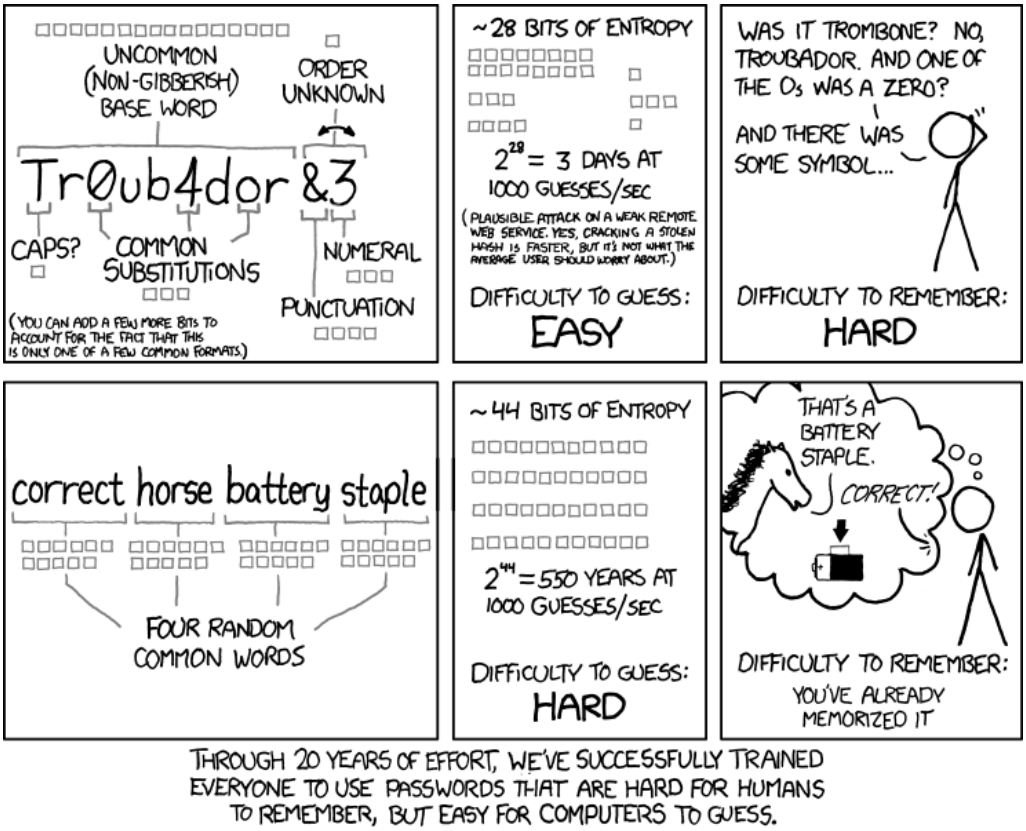
Le mot de passe est un élément « que je connais ».

Une politique de sécurité doit imposer des règles pour leur constitution, et elle doit les rendre suffisamment solides.

Exemple de politique :

- Au moins 10 caractères (en 2016). Il ne faudrait plus parler de « mots de passe », mais de « phrases de passe » (pour faciliter la mémorisation) ;
- Au moins une majuscule ;
- Au moins une minuscule ;
- Au moins un caractère spécial (de ponctuation) ;
- Pas plus de deux caractères identiques consécutifs).

Attention cependant aux politiques trop complexes qui rendent les mots de passe trop compliqués à retenir et donc poussent les utilisateurs à les écrire sur des post-it, ou à multiplier les appels au support ou aux fonctions « oubli de mot de passe » pour les renouveler.



(https://xkcd.com/936/)

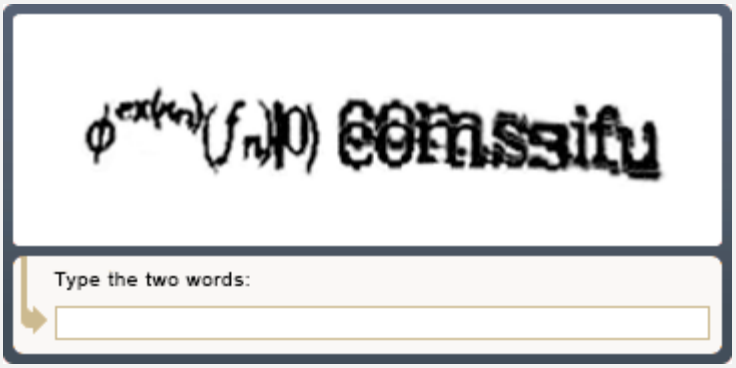
D’autres règles peuvent être imposées par la politique de sécurité, comme l’interdiction de réutiliser d’anciens mots de passe ou des parties d’entre eux.

Toutes les vérifications de conformité doivent se faire au niveau du serveur, comme n’importe quelle procédure de validation de données, puisque le côté client est à la main d’un éventuel attaquant. Il est éventuellement possible de pousser certaines de ces vérifications côté client, afin de fournir un retour immédiat à l’utilisateur, mais attention aux problèmes de contraintes différentes à satisfaire simultanément…

Les mots de passe / phrases de passe ne doivent jamais être stockés en base de données ou en dur : ce sont leurs condensés (*hash*) qui doivent l’être.

Question	Qu’est-ce que l’on perd en stockant les mots de passe de façon dérivée (condensée / *hash*) ?
Réponse	La possibilité de les fournir à l’utilisateur lorsqu’il les oublie… qui reste de toute façon une mauvaise idée sans canal sécurisé.

	Nous sommes à nouveau dans la notion de compromis. Par exemple : pourquoi les systèmes Microsoft sont vulnérables au *pass-the-hash* ? C'est un compromis entre l'avantage de pouvoir rejouer de façon transparente une authentification et le risque de ne pas se réauthentifier.

Question	À quoi servent les *captcha* ? 
Réponse	À s'assurer qu'il y a bien un humain derrière une authentification, c'est-à-dire qu'il ne s'agit pas d'un automate.

Question	Pourquoi ajouter des aléas aux mots de passe avant de les condenser ?
Réponse	Pour rendre plus complexe leur découverte en cas de compromission de la base de hash par exemple. Les *rainbow table* ne fonctionneront pas.

On peut casser des mots de passe dans deux circonstances bien différentes :

- En ligne, et alors il y a en général des restrictions : blocage de compte après un certain nombre d'essais, temporisation, détection des tentatives… ;
- Hors-ligne : mais il faut avoir réussi à récupérer un compte (par exemple en l'interceptant) ou une base de comptes (comme un

/etc/shadow ou passwd sur UNIX / Linux ou une base SAM locale sur Windows). L'avantage est alors qu'on dispose de tout son temps et que l'on peut déployer toute la puissance de calcul à disposition.

Méthodes hors ligne: Dictionnaire, *Rainbow table*, force brute, force brute avec masque

Quelques chiffres de temps de cassage, fin 2016 :

- Hash d'un mot de passe NTLM de 8 caractères avec utilisation d'une *Rainbow table*: 9 heures ;
- 10j avec une carte Tesla (équivalent à une carte graphique grand public) ;
- Hash UNIX DES (salé) de 8 caractères avec une carte Tesla: 4 ans et 200j (salage : ajout d'une donnée aléatoire au mots de passe afin que deux informations identiques aient une empreinte différente – utilisé pour complexifier les attaques par analyse de fréquences ou par *rainbow table*, comme https://crackstation.net/) ;
- MD5 de 8 caractères: Temps max: 17j avec la Tesla ;
- Hash NTLM de 8 caractères (minuscules seulement): 30s avec la Tesla ;
- Hash NetNTLMv2 de 8 caractères : 1 an et 317j avec la Tesla ;
- Hash NetNTLMv2 de 8 caractères avec un masque: 6h avec la Tesla.

Pour stocker et créer des mots de passe, utilisez un outil comme KeePass. Cet outil pourra aussi vous donner une idée de la qualité d'un mot de passe :

https://keepass.info/help/kb/pw_quality_est.html

Enfin, vérifiez régulièrement https://haveibeenpwned.com/

XII. Détection

Deux principes vont éclairer ce chapitre :

> La prévention c’est l’idéal, la détection est nécessaire.

> On ne peut pas se prémunir de toutes les menaces.

Mais il faut essayer de mettre en place une sécurité qui permette de détecter les attaques, ou au moins de disposer de données pour analyse a posteriori. Si on sait ce qui s’est passé au cours d’une attaque, on peut espérer empêcher que cela n’arrive à nouveau.

La capacité opérationnelle de détection de tentatives – réussies ou non – d’intrusions ou d’attaques est essentielle : quelles que soient les mesures de sécurité mises en place, il doit y avoir des vigies.

Cette capacité opérationnelle repose sur des techniques, des processus et une organisation.

Un aspect essentiel, le partage d’informations avec les pairs :

> Partagez avec vos pairs. Ma détection est votre prévention.

Traces, détections, contrôles, alertes

Les systèmes, équipements, logiciels et applications génèrent des traces, des logs, des journaux d’évènements.

Les traces sont utiles pour développer des programmes, pour surveiller leur bon fonctionnement, mais aussi pour repérer des problèmes de sécurité.

Souvent, après un incident de sécurité, on en voit les traces dans les logs.

L’enjeu est de réduire le temps de détection, si possible jusqu’à pouvoir alerter en temps réel.

L’analyse *ex post* des traces permet aussi d’effectuer des contrôles de conformité (avec des informations sur les états).

Enfin, des traces fiables sont essentielles pour effectuer des activités d'analyse *post-mortem*, de *forensics*.

Les alertes sont définies lorsque certaines conditions sont réunies. Quelques exemples :

- Plus de 10 tentatives d'authentification en erreur en moins de 1 minute ;
- Si on est capable d'identifier les machines d'administration : toute connexion SSH ou RDP aux serveurs ne provenant pas de ces machines ;
- Ajouts dans des groupes de comptes à haut privilège (stables usuellement) ;
- Connexions aux interfaces d'administration hors des horaires de travail ;
- Détection des WAF ou sondes IDS.

Les alertes peuvent être envoyées par *traps* SNMP, courriels, SMS…

Une fois que l'on dispose des traces et éventuellement d'alertes, le vrai travail commence : les interpréter.

SIEM (*Security Information and Event Management*)

Les SIEM sont des logiciels qui agrègent des traces provenant de diverses sources, afin de détecter des problèmes de sécurité, d'effectuer des recherches et des contrôles. Ses principales fonctions sont :

- Agrégation de traces provenant de sources différentes ;
- Fonctions de corrélation ;
- Génération d'alertes ;
- Génération de tableaux de bord ;
- fonctions de recherche et d'analyse post-mortem (*forensics*).

Quelques exemples de SIEM commerciaux : Splunk, QRadar, ArcSight…

Exemple de SIEM *open source* : ELK (Elasticsearch / Logstash / Kibana).

Scanners

Un scanner est un logiciel qui permet de détecter et caractériser des vulnérabilités sur des systèmes, applications, équipements réseau, etc.

Exemples :

- nmap –v [nom de domaine] (scan basique, ports résevés)
- nmap 192.168.1.0/24 (scan d'un sous-réseau)
- nmap -O [cible] (détection d'OS)
- nmap -p 1-65535 -sV -sS -T4 [cible] (sS : furtif ; sV : détection de version ; T4 : niveau « d'agressivité »)

Un scanner de port va identifier les ports ouverts sur un serveur et tenter d'identifier les services qui tournent derrière (exemple : nmap), un scanner de vulnérabilités (exemple : Nessus) va par exemple permettre de détecter :

- Des défauts de configuration présentant des risques (comme des services inutiles, vulnérables, des mots de passe par défaut, etc.) ;
- L'absence de correctifs de sécurité ;
- Des vulnérabilités sur le système cible qui permettent à un attaquant de lancer des attaques à distance ;
- Des vulnérabilités aux dénis de service (DoS ou DDoS) ;
- Des audits spécifiques (comme PCI-DSS).

Des scans devraient être réalisés régulièrement. Si c'est le cas, ils peuvent permettre de suivre l'évolution du niveau de sécurité au fil du temps.

Les scanners doivent être mis à jour régulièrement, et configurés finement.

Leurs résultats peuvent être couplés avec les journaux d'évènements pour :

- Vérifier que les scans sont bien détectés ;
- Pouvoir corréler des détections d'attaques avec les vulnérabilités avérées (que l'on sait exister au sein de notre SI).

Quatre outils typiques : nmap, Nessus, Burp, ZAP.

Tests de sécurité

Question	Pouvez-vous citer quelques exemples de vulnérabilités ?
Réponse	Exemple de XSS. Pour tester, se référer à l'OWASP. Par exemple : <script>alert(Bonjour ! Vulnérable XSS !)</script> ou "><script>alert(document.cookie)</script> Si vous pensez qu'il y des filtrage sur les <, essayez : %3cscript src=http://www.example.com/malicious-code.js%3e%3c/script%3e Pour une collection mise à jour quotidiennement de sites vulnérables à des XSS : https://www.xssed.com. Ne semble plus maintenu.

L'intérêt des tests de sécurité peut se résumer en deux devises :

> Découvrez vos propres faiblesses avant qu'un tiers ne le fasse.

> La connaissance des attaques oriente la défense.

Les tests de sécurité vont non seulement permettre de découvrir des vulnérabilités à corriger, mais ils vont aussi aider à définir des mesures de sécurité préventives réduisant la probabilité d'exploitation de futures vulnérabilités.

Il y a plusieurs types de tests de sécurité :

- Les scans, authentifiés ou non ;
- Les audits de configuration ;
- Les tests d'intrusion en « boite noire » ;

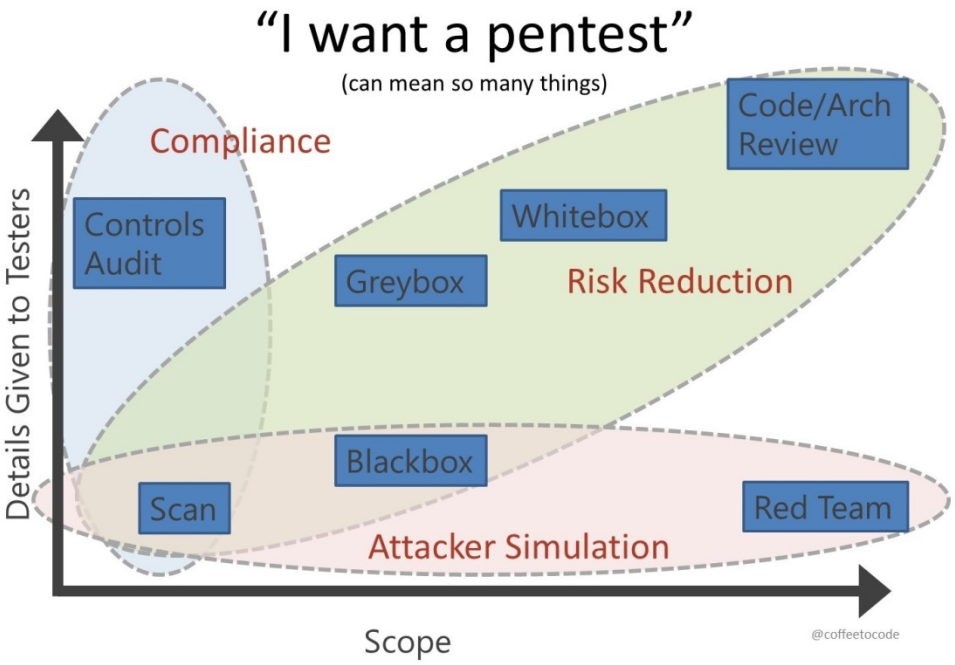
- Les tests d'intrusion en « boite blanche » ;
- Les tests en « boite grise » ;
- Les audits de code.

En mode « boite noire » (*black box*), le *pentester* ne dispose d'aucune information sur l'infrastructure ou les détails de l'application, ni de comptes d'accès sur la cible (il ne dispose en général que de l'adresse). C'est la méthode qui simule au plus près une attaque réelle, mais elle permet souvent d'identifier moins de vulnérabilités qu'en « boite blanche » ou « grise ».

En mode « boite blanche » (*white box*), le p*entester* dispose de toutes les informations qu'il souhaite sur la cible à tester, comme des comptes à privilèges, le code source disponible, les schémas d'architecture, etc.

Le mode « boite grise » (*grey box*) est un intermédiaire entre ces deux derniers modes. Le *pentester* dispose de certaines informations (comme des schémas d'architecture, une démonstration de l'application, la possibilité de poser des questions, etc.), des comptes valides…

Le détail du mode choisi, de la méthodologie, des responsabilités est défini dans un protocole d'accord signé avant le début des tests.



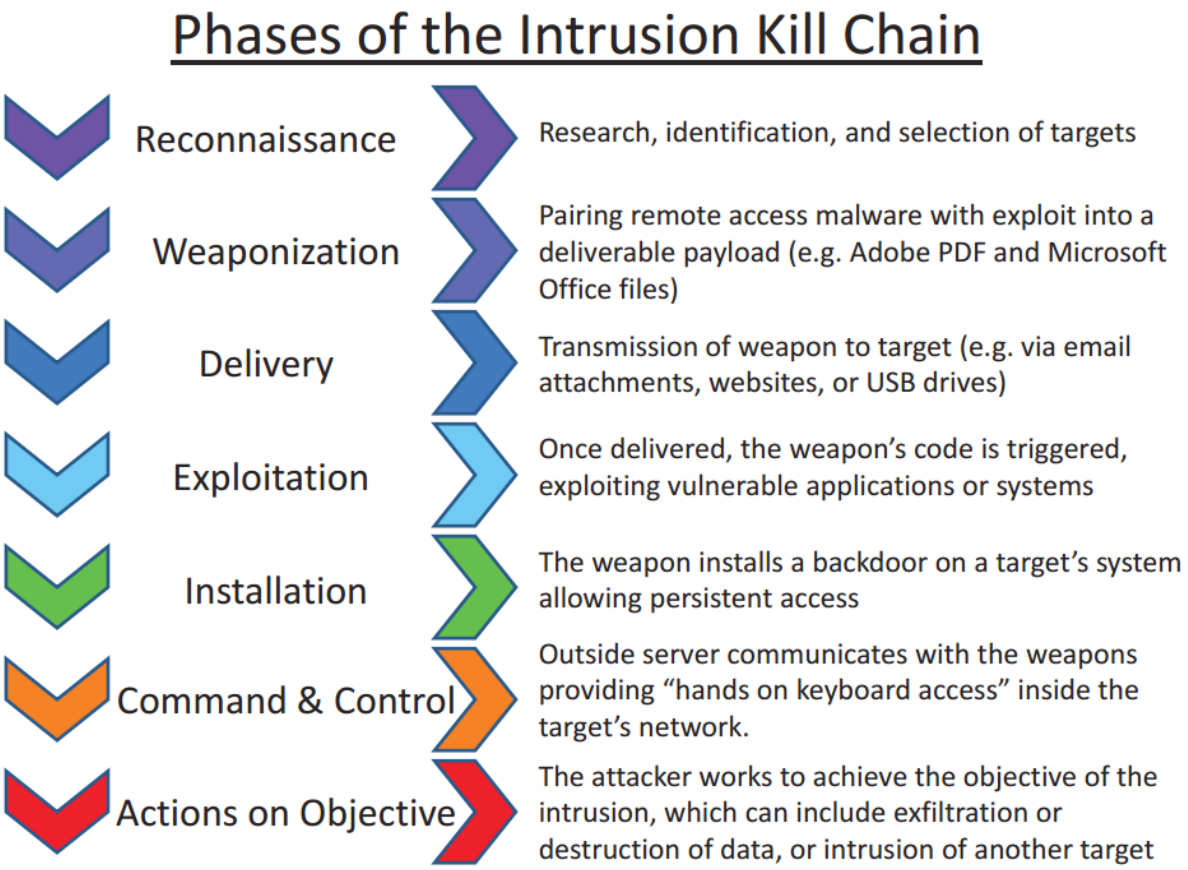
La qualité d'un test de sécurité dépend essentiellement de la compétence et de l'expérience des analystes.

Les tests d'intrusion vont consister à identifier des vulnérabilités et tenter de les exploiter.

Les méthodologies typiques de réalisation des tests d'intrusion simulent la réalisation d'une vraie attaque :

- Reconnaissance ;
- Scanning ;
- Obtention d'accès ;
- Persistance des accès ;
- Exploitation ;
- Couverture des traces.

La *Cyber Kill Chain* :



(source : https://en.wikipedia.org/wiki/Kill_chain)

Une méthodologie de *pentest* va ajouter en amont une phase initiale de préparation (avec, en particulier, définition et signature du protocole d'accord), et en aval une phase d'analyse et de reporting.

Phases de *pentests* applicatifs :

- **Reconnaissance** : recherche de la cible, diagramme de la cible, recherches d'informations pour casser des mots de passe… ;
- **Recensement** : caractérisation des cibles et de ce qui l'entoure, scan de port, caractérisation des OS, relations entre les applications… ;
- **Découverte** : recherche de vulnérabilités, premiers flux malveillants, focalisation et rédaction de scripts – si nécessaire : applications les plus courantes, interfaces utilisateur, révélation de données, systèmes d'authentification, messages d'erreur… ;
- **Exploitation** : lancement des attaques…

Méthode itérative et cyclique : dès qu'une vulnérabilité est exploitée, on recommence depuis le début.

Pour la reconnaissance, ça peut commencer par des recherches sur des informations publiques, « ouvertes ».

Si ce domaine vous intéresse et que vous voulez vous exercer

Exercices en ligne:

https://tryhackme.com/

https://www.root-me.org/

https://www.newbiecontest.org/

DVWA : http://www.dvwa.co.uk/

WebGoat : https://owasp.org/www-project-webgoat/

Gruyere : https://google-gruyere.appspot.com/

Google dorks

Opérateurs de recherche standard :

- "" (chaîne de caractère complète)
- OR (opérateur ou)
- AND (opérateur et)
- - (opérateur qui permet d'exclure)
- * (joker. Exemple : "le meilleur * de paris")
- () (permet de grouper des termes. Exemple : (master OR licence) "paris XIII"

Opérateurs de recherche avancés:

- site : domaine.com (dans le site...)
- inurl : phpinfo (qui contient dans l'URL ...)
- intitle : "Admin login" (dont le titre de la age contient...)
- cache : ... (dans le cache de Google...)
- filetype : xls (dont l'extension est...)

Exemples de recherches :

- filetype:log inurl:auth.log -site:github.com
- inurl:/host.txt + filetype:txt + "password"
- intitle:index.of .bash_history

Plus d'information sur les « *Google dorks* » : https://www.exploit-db.com/google-dorks

Mais attention à ce à quoi vous accédez.

Des nombreux outils aident à réaliser les tests de sécurité.

Proxy web : Burp…

Scanner : nmap, Nessus, ZAP…

Exploitation : Metasploit…

Distribution Linux : Kali

XIII. Réaction

Gestion des incidents de sécurité

Il s'agit des procédures et des plans d'action permettant de gérer les incidents de sécurité comme :

- Les infections par des *malwares* ;
- Les vols de données ;
- Les intrusions (comme les APT – *Advanced Persistent Threat*) ;
- Les dénis de service ;
- Les usurpations d'identité ou d'image ;
- …

Pour les gens de la sécurité, dans la plupart des cas, « incident » est entendu comme malveillance, ou volonté de malveillance.

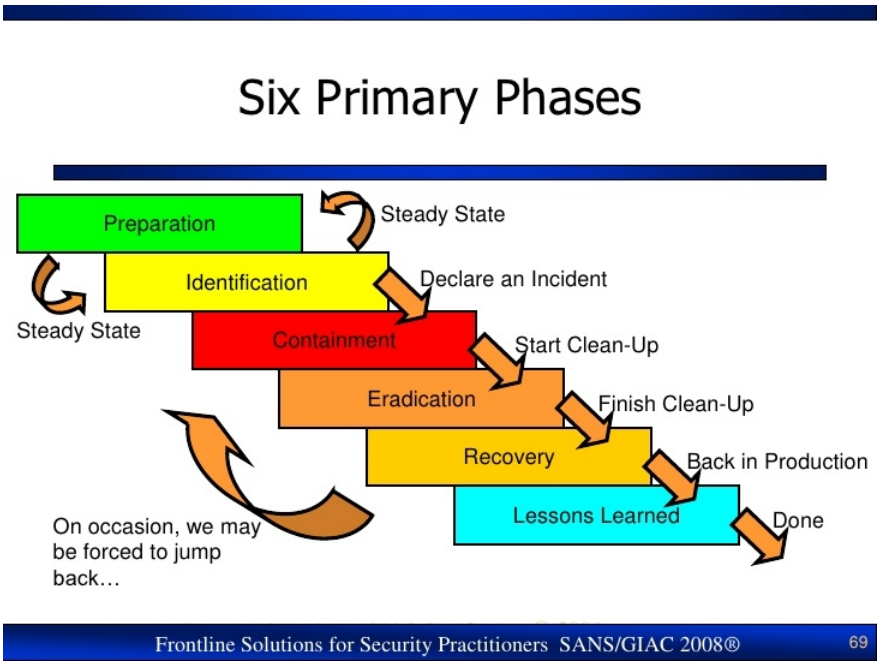
Les procédures de réaction sont comme les sauvegardes : vous ne les utiliserez pas souvent, mais quand vous en aurez besoin, vous serez content de les avoir.

En anglais, le terme communément utilisé pour désigner les opérations de réponse à incident et d'investigation est *Digital Forensics & Incident Response (DFIR).*

Les phases de la gestion d'incident :

- **Préparation** :
  - Organisation des équipes ;
  - Procédures et politiques (de confidentialité en particulier) ;
  - Outillage (par exemple des moyens de communication indépendants du réseau) ;
  - Préparation de la communication.
- **Identification** :
  - Détection des incidents via les alertes, l'analyse des traces, des signalements des utilisateurs, recherches sur la base d'IoC (*Indicators of Compromission* : signatures et condensés de *malware*s, @IP et URI malveillantes, domaines et adresses de C2, etc.).
- **Endiguement** :
  - Stopper, limiter les dégâts ;
  - Synthétiser la situation, informer les responsables.
- **Élimination** :
  - Suppression des vecteurs de l'attaque et premières analyses des causes ;
  - Restauration à partir de sauvegardes saines ;
  - Modification des configurations sécurité si nécessaire.
- **Retour à la normale** :
  - Remise des systèmes en production ;
  - Validation de l'intégrité des systèmes et des services ;
  - Surveillance d'évènements suspects.
- **Enseignements** :
  - Rédaction d'un rapport ;

- Proposition d'amélioration des mesures de sécurité préventives ;
- Amélioration de la procédure de gestion d'incident.



D'après le SANS, les 7 péchés capitaux de la réaction suite à incident de sécurité, priorisés:

1. Absence de signalement ou de demande d'aide ;
2. Notes absentes ou médiocres ;
3. Mauvaise manipulation ou destruction des preuves ;
4. Échec de la restauration à partir des sauvegardes ;
5. Échec de l'endiguement ou de l'élimination ;
6. Prévention de la réinfection en échec ;
7. Échec de l'application des enseignements.

En synthèse :

> Il faut être prêt à réagir. Et cela ne s'improvise pas.

Gestion de crise

Au sein de toute organisation devrait exister une procédure simple et fonctionnelle permettant de convoquer et tenir des cellules de crise.

Caractéristiques des cellules de gestion de crise :

- Doit pouvoir être convoquée rapidement ;
- Doit mettre en présence les dirigeants capables de prendre des décisions immédiates et les techniciens les plus qualifiés et impliqués dans la gestion de l'incident.

Responsabilités :

- Décider d'un plan d'action (pour résolution de l'incident et traitement du / des problèmes) ;
- Gérer la communication.

Remarques importantes :

- **Séparation des rôles** : les propositions de solutions devraient venir des techniciens, pas des dirigeants. Si le mode de décision n'est pas autogestionnaire, ces derniers doivent assumer leurs responsabilités et faires des choix clairs ;
- **La hiérarchie doit se garder de peser sur les techniciens**, en particulier lorsque ces derniers travaillent – syndrome du chef qui regarde par-dessus l'épaule ; des points de situation doivent être régulièrement organisés, mais il faut éviter de perturber ceux qui essaient de résoudre les problèmes ;
- **Entraînement :** de telles cellules de crises ne peuvent fonctionner que si l'organisation s'exerce à les convoquer et joue des scénarios en guise d'entraînement (exercices sur table).

Organisation de détection et réaction, SOC et CSIRT :

Un SOC (Security Operation Center) assure la supervision des systèmes d'information, dans toutes ses dimensions et fonctions (applications, serveurs, bases de données, équipements réseaux, services externalisés, postes de travail et autres objets connectés)

Composé d'analystes sécurité, ses principales activités sont de:

- collecter des événements de sécurité ;

- analyser et détecter des anomalies sécurité ;

- définir et mettre en œuvre des interventions pour endiguer les attaques.

Ces actions reposent sur des outils, le premier d'entre-eux étant le SIEM. Sur les postes de travail, l'utilisation de fonctions apportées par des EDR (Endpoint Detection and Reaction) peuvent être très utiles.

Les détections d'anomalies relèvent de trois grandes catégories:

- les manquements aux politiques de sécurité, aux règles d'hygiène informatique;

- les attaques, *phishing*, *spear phishing*, tentatives d'intrusion, APT, rançongiciels, etc.;

- les incidents ou quasi-incidents « métier », c'est à dire ceux liés à des règles de sécurité dépendant de l'activité de l'entité.

Les CSIRT (Computer Security Incident Response Team), ou CERT (qui est un nom déposé), sont en principe plus axés sur la réaction et surtout sont plus impliqués dans des démarches « communautaires » (liens avec d'autres équipes). Mais les différences sont souvent ténues.

XIV. Éthique et légalité

Il faut toujours penser ce que l'on fait, du point de vue de la légalité comme de celui de l'éthique.

Légalité

Il y a des actions légales et d'autres illégales.

Attention lorsque vous menez des tests.

En particulier, l'informatique traite souvent des données personnelles et il existe des lois, décrets et jurisprudences qui régissent ce qu'il est possible de faire, et sous quelles conditions, et ce qu'il n'est pas possible de faire en traitant des données personnelles.

En France, la loi fondatrice est la loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés.

Les cinq **principes fondateurs** en sont :

- **Finalité** : les données à caractère personnel recueillies et traitées ne doivent l'être que pour un usage légitime et déterminé, préalablement défini ;
- **Pertinence des données** : seules les données pertinentes et nécessaires au regard des objectifs poursuivis doivent être traitées ;
- **Conservation des données** : la durée de conservation doit être déterminée en fonction de la finalité du traitement ;
- **Obligation de sécurité** : obligation de mettre en œuvre des mesures de sécurité pour protéger les données, en particulier leur intégrité et confidentialité ;
- **Respect du droit des personnes** : les personnes doivent disposer de droits leur garantissant la maitrise des informations les concernant. Elles doivent notamment être informées, doivent pouvoir s'opposer au traitement de leurs données (sauf autorisation légale de traiter ces données) et doivent avoir un droit d'accès et de rectification aux données les concernant.

Quiconque met en œuvre des traitements de données personnelles est tenu de le déclarer à la Commission Nationale Informatique et Libertés (CNIL).

Site de référence : https://www.cnil.fr

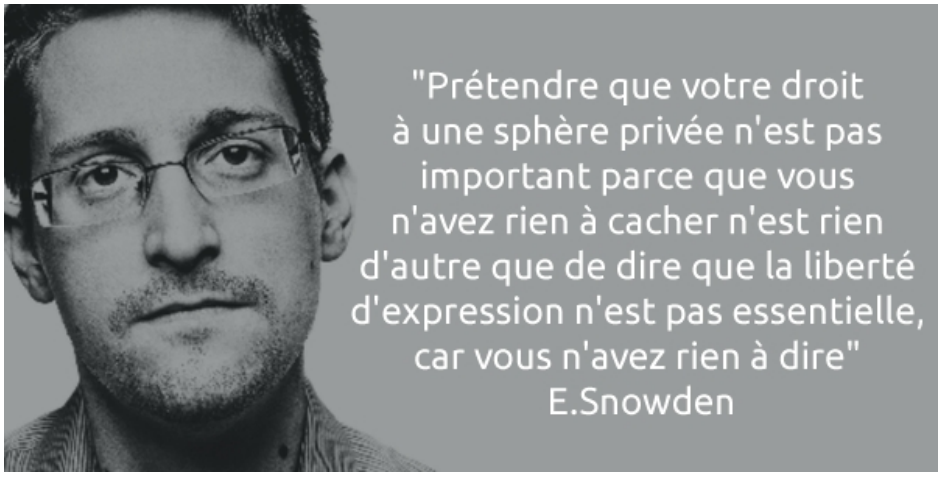
Dans certaines grandes organisations, il existe une fonction en charge d'effectuer les déclarations et de veiller au respect des obligations en matière de protection des données personnelles : le Correspondant Informatique et Libertés (CIL).

Les activités de tests de sécurité (*pentest*) doivent également être très encadrées par des accords explicites, pour ne pas risquer de tomber sous le coup du Code Pénal et des jurisprudences concernant les intrusions informatiques.

Le « maintien frauduleux dans un système de traitement informatique de données » tombe sous le coup de la Loi Godfrain du 5 janvier 1988.

En cas de découverte d'une vulnérabilité majeure, et si l'on veut la signaler, certaines précautions doivent également être prises.

Éthique



Au-delà de la légalité des actions, il est vital de toujours :

> Pensez ce que vous faites en termes éthiques, pour qui, pour quelle finalité.

D'autant plus que les informaticiens, et dans la sécurité en particulier, ont souvent de grands pouvoirs dans notre société de plus en plus numérisée, électronique, connectée, digitale, cyber-truc… choisissez le *buzzword* qui vous plait.

Or :

Avec de grands pouvoirs viennent de grandes responsabilités.

Les individus peuvent renoncer à formuler des jugements éthiques et moraux sur leurs actions ou celles qui les entourent, se contentant d'obéir aux instructions ou aux procédures.

Ils cessent de penser, démissionnent et ne se voient plus que comme un rouage qui n'a pas son mot à dire (cf. le concept de banalité du mal d'Hannah Arendt).

Ces renoncements à penser par soi-même peuvent être des facteurs clés dans la commission d'actions immorales (Exemple funeste : Amesys, qui « a vendu des technologies de surveillance qui ont permis au régime du colonel Kadhafi de surveiller, de traquer et, à terme, d'éliminer ses opposants et d'organiser le monitoring de l'ensemble des communications sur les réseaux Internet, téléphonie mobile et satellite en Libye » (Wikipédia).

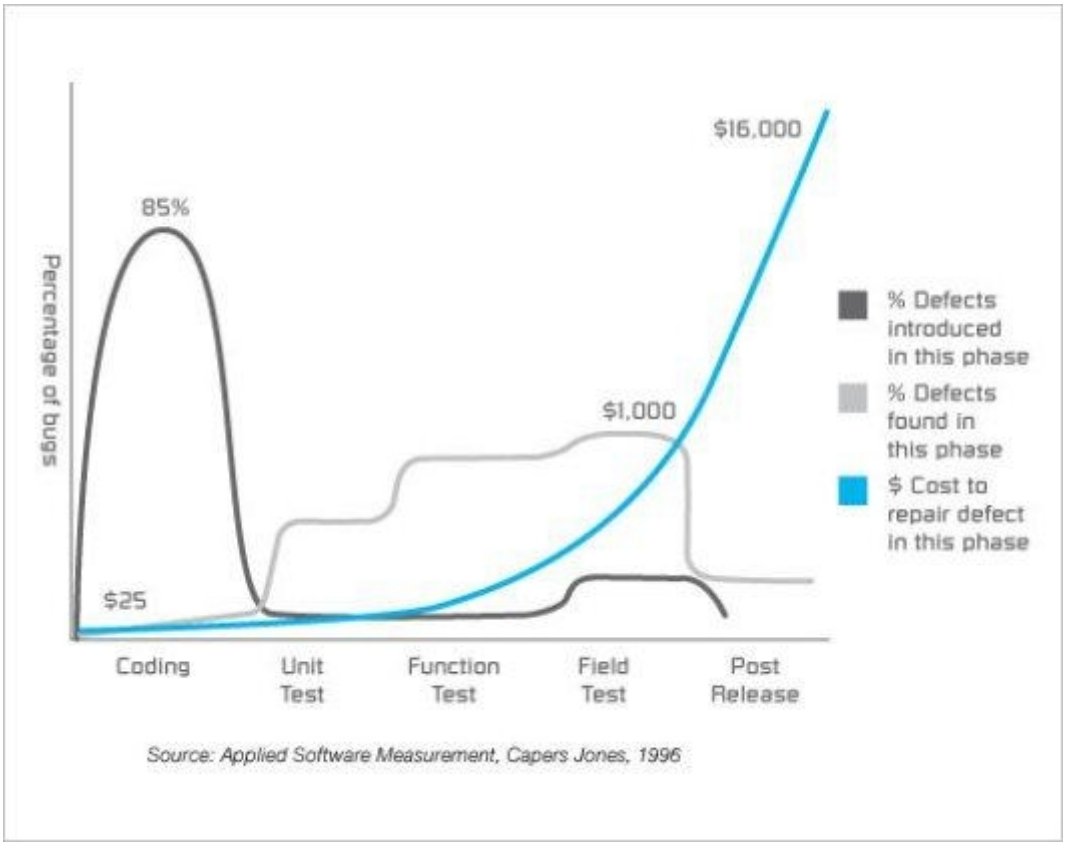
La fin ne justifie pas les moyens. Les moyens déterminent la fin.

XV. Sécurité des applications

Avant de commencer vos développements – après avoir recueilli les besoins des clients – mais avant de commencer à spécifier :

Faites des analyses de risques. Identifiez les menaces, les enjeux, les données et les services critiques et déduisez-en des exigences de confidentialité, intégrité et disponibilité.

Plus tard une vulnérabilité est découverte et corrigée, plus elle est couteuse (en risque, en temps, en argent…).



Les développeurs sont les acteurs clés de la sécurité des applications. Il ne faut pas imaginer qu'un WAF, que l'utilisation de modules ou bibliothèques externes ou qu'un quelconque produit va tout résoudre. Souvenez-vous de l'exemple des XSS : la résolution est bien moins coûteuse à la source.

Souvenez-vous que :

> Le produit hérite de ses processus de production.

Top 10 de l'OWASP

L'OWASP Top 10 est un document listant les 10 types de vulnérabilités applicatives les plus critiques et les plus souvent rencontrées.

La liste se base sur des statistiques de centaines de milliers de vulnérabilités effectivement constatées.

Ce document, qui est essentiellement un outil de sensibilisation, est destiné aux développeurs, aux architectes de solution, aux décideurs, aux auditeurs…

La version actuelle est la « 2017 ».

https://owasp.org/www-project-top-ten/

https://owasp.org/www-project-top-ten/2017/

A1 - Injection

Les vulnérabilités par injection arrivent lorsqu'une application accepte des données non sûres dans une commande ou une requête et qu'un interpréteur les traite. Les données malveillantes peuvent alors faire exécuter des actions non prévues à l'interpréteur ou accéder à des données sans authentification.

Exemple d'injections : SQL, OS, LDAP, XML, SMTP…

Exemples de mesures de sécurité :

- Si possible, utilisez des API avec une interface paramétrable avec des requêtes paramétrées, des procédures stockées (base de données)… ;
- Sinon, échapper les caractères spéciaux ;
- Si possible, valider les données utilisateur en mode liste blanche ;
- Restreindre les privilèges d'accès aux bases de données.

http://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

A2 - Violation de gestion d'authentification et de session

Mauvaise gestion des fonctions d'authentification et de gestion des sessions permettant de compromettre des mots de passe, des tokens de sessions ou d'usurper des identités

Exemples :

- Identifiants de session dans l'URL ;
- Identifiants de sessions prédictibles ;
- Authentification ou cookie de session envoyé en HTTP ;
- Stockage des mots de passe en clair ;
- Fonctions de récupération des mots de passe oubliés faible.

Exemples de mesures de sécurité :

- Utiliser les mécanismes standards de gestion de session des langages et *frameworks* de développement ;
- Connexions des utilisateurs et identifiants de sessions doivent transiter chiffrés (TLS) ;
- Stocker les mots de passe des utilisateurs uniquement de façon condensée.

https://www.owasp.org/index.php/Authentication_Cheat_Sheet

A3 - Cross Site Scripting (XSS)

Un XSS permet d'exécuter du code JavaScript dans le contexte du navigateur de l'utilisateur. La cause en est un mauvais filtrage des entrées de l'application. Ils peuvent être « réfléchis » (attaque via un lien malveillant) ou « stockés » (le code malveillant est déposé sur l'application).

Ce n'est pas une vulnérabilité anodine. Grâce à un outil comme BeEF il est possible d'accéder à des cookies de sessions, à l'historique du navigateur, passer des commandes au navigateur, etc. et exploiter d'autres vulnérabilités.

Exemples de mesures de sécurité :

- Échapper les données non fiables ;
- Lorsque possible, mettre les entrées en liste blanche ;

- Utiliser des librairies de nettoyage comme AntiSamy d'OWASP ou Java HTML Sanitizer Project.

http://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

A4 - Références directes non sécurisées à un objet

Exposition d'un objet interne (fichier, répertoire, clés, etc.) sans contrôle d'accès

Exemples de causes :

- Contrôle effectué sur la base d'un identifiant prédictible (identifiant, nom de fichier, etc.) ;
- Absence de contrôle d'accès aux ressources.

Exemples de mesures de sécurité :

- Implémenter des références indirectes par user ou par session (à l'aide de la bibliothèque ESAPI par exemple) ;
- Mettre en place un contrôle d'accès pour toutes les ressources.

A5 – Mauvaise configuration sécurité

Des paramètres de sécurité doivent être définis et appliqués sur les applications, *framworks*, serveurs d'applications, serveurs web, bases de données…

Exemples de mesures de sécurité :

- Durcissement des systèmes (par exemple : modification des paramètres par défaut – Drupal / Joomla / Typo3… - désactivation / suppression de services inutiles, restriction de droits, non-affichage des messages d'erreur), que ce soit en production ou sur les autres environnements (et mettre des mots de passe différents sur ces environnements) ;
- Mise à jour des applications, systèmes, librairies (*patch management*) ;
- Dérouler régulièrement des scans pour détecter des problèmes de configuration sécurité ou des correctifs de sécurité manquants.

(https://www.blackhat.com/docs/us-16/materials/us-16-Thomas-Can-You-Trust-Me-Now.pdf)

Question	Que pensez-vous de cette mesure de sécurité (sur la photo) ?
Réponse	Vraiment ? Avez-vous déjà fait de la moto (ou du scooter) avec une roue entravée par une chaine ? Avoir passé la chaine autour du plot peut tout simplement avoir été fait pour éviter la pisse des chiens… Moi je l'aurais fait. Cette photo illustre une trivialité : on ne peut être bon partout. Celui qui a pris cette photo et mis ce titre est un très bon expert sécurité informatique; mais pas forcément un bon expert en sécurisation de motos.

A6 – Exposition de données sensibles

Stockage non sécurisé de données confidentielles (numéros de carte de crédit, données personnelles, d'authentification…). Risque de vol ou de modification.

Quelques exemples :

- Authentification ou cookies transmis en HTTP simple ;
- Absence de chiffrement – ou mauvais chiffrement – des données sensibles.

Exemples de mesures de sécurité :

- Commencer par identifier les données sensibles, se poser la question de leur sécurité ;
- Ne pas stocker les données non nécessaires ;
- Choisir de bonnes méthodes de chiffrement (algorithmes robustes – ne pas en créer, clés suffisamment fortes, bonne gestion des clés) ;
- Stocker les mots de passe de façon dérivée, condensée, en utilisant des algorithmes éprouvés (comme Bcryp, PBKDF2, scrypt).

(https://www.blackhat.com/docs/us-16/materials/us-16-Thomas-Can-You-Trust-Me-Now.pdf)

A7 – Manque de contrôle d'accès au niveau fonctionnel

Mauvais contrôle d'accès aux fonctions

Typiquement, ce peut être une fonction invisible sur l'IHM mais accessible en devinant un lien ou un paramètre. Idem pour les pages.

Exemples de mesures de sécurité :

- Appliquer par défaut le principe : tout ce qui n'est pas explicitement autorisé est interdit / soumis à des droits d'accès ;
- Lorsque nécessaire, restreindre les fonctions aux utilisateurs authentifiés ;

- Vérifier les droits des utilisateurs lorsqu'ils accèdent à une ressource, quelle que soit la façon dont cette ressource est appelée.

A8 – Falsification de requête intersite (CSRF – Cross Site Request Forgery)

L'attaquant effectue une action à l'insu de l'utilisateur, par exemple s'il est possible de prévoir les actions possibles sur une application. L'attaquant peut alors forcer le navigateur à émettre une requête en lieu et place de l'utilisateur (lien dans un courriel, balise image, XSS)

Exemple : consulter la liste des utilisateurs

Exemples de mesures de sécurité :

- Utilisation d'un *token* unique à chaque requête. Une possibilité est d'inclure ce *token* dans un champ caché (*hidden*) : il ne passera pas par l'URL, ce qui est la méthode d'attaque la plus courante ;
- Imposer aux utilisateurs de confirmer leur volonté sur les fonctions sensibles (par exemple CAPTCHA ou réauthentification) ;
- Projet OWASP CSRF Guard.

http://www.owasp.org/index.php/CSRF_Prevention_Cheat_Sheet

A9 - Using components with known vulnerabilities

Les composants comme les librairies, les *frameworks*, CMS, les modules logiciels qui ont des vulnérabilités peuvent être la cible d'attaques, qui seront d'autant plus graves que les droits qui les font tourner sont élevés.

Exemple de mesure de sécurité :

Identifier tous les composants, surveiller la publication d'annonces de vulnérabilités et les mettre à jour.

A10 – Redirections et renvois non validés

Si les redirections et renvois vers d'autres pages reposent sur des paramètres non validés côté serveur, un attaquant peut les modifier.

Exemples de mesures de sécurité :

- De préférence, ne pas utiliser renvois et redirections ;
- Si c'est indispensable : ne pas spécifier la destination dans les paramètres utilisateur, ou alors valider la valeur de cette destination côté serveur.

(Traduction en français de la version 2013, Guillaume Lopes : https://www.ossir.org/paris/supports/2014/2014-01-14/OSSIR-Paris-Intrinsec-OWASP-Top-Ten-2013.pdf)

N'en restez pas au Top 10

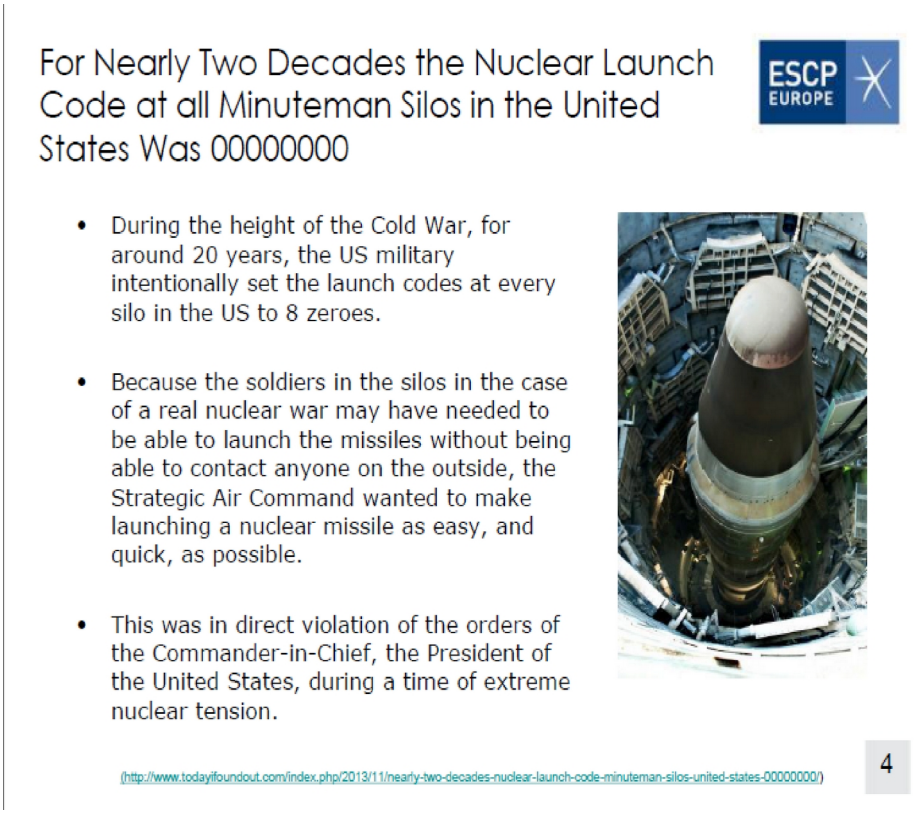
Ce ne sont « que » les vulnérabilités les plus critiques.

Mais il peut exister bien d'autres problèmes de sécurité applicative que ce « Top 10 » n'évoque pas.

Par exemple :

- Ne faites pas de vérification sécurité côté client, en particulier de droits d'accès ;
- Ne laissez pas d'informations sensibles dans les commentaires de votre code public (JS, HTML, etc.) ;
- Ne mettez pas en ligne en libre accès des applications en test / en développement ;

- Dédiez des chapitres spécifiques « sécurité » dans les cahiers de recette, en particulier afin de concevoir et dérouler des tests non passants ;
- Anonymisez les données confidentielles avant de les passer de la production à des environnements moins sécurisés, par exemple pour des tests ;
- S'il s'agit d'une application critique avec un haut niveau de sécurité, prévoyez toujours une procédure de débrayage des mesures de sécurité, en cas d'urgence. Il doit être possible de débrayer la sécurité pour faire fonctionner le système ;
- …



Utilisez les guides de développement sécurisé de l'OWASP.

https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/migrated_content

https://owasp.org/www-project-cheat-sheets/

Il est possible de réaliser un audit de code, manuel ou à l'aide d'outils (par exemple https://www.sonarqube.org/)

Écrivez ce que vous faites : essayez d’être le plus rigoureux possible. Et si des actions sont amenées à se répéter, définissez un processus formel.

N’oubliez pas :

> Le produit hérite de ses processus de production.

XVI. Récapitulatif : les grands principes de la sécurité

Politique de sécurité

La sécurité globale d'un système est celle du maillon le plus faible. Les efforts de sécurisation doivent être homogènes.

Faites des analyses de risques, même minimales, en tenant compte des menaces, identifiez les données et services critiques et déduisez-en des exigences de confidentialité, intégrité et disponibilité.

Classifiez vos données (publique, restreinte, confidentielle). Identifiez les données et processus sensibles.

Il est impossible d'interdire ce que vous devez autoriser. Et donc, sachez accepter ou faire accepter les risques identifiés.

Écrivez ce que vous faites, faites ce que vous écrivez.

Toute solution de sécurité mal utilisée, mal configurée, peut lourdement handicaper l'expérience utilisateur.

Ne pensez pas que par déductions et inductions. La pensée par analogie est souvent très fructueuse.

Principes opérationnels

À mesure que la sophistication des usages croit, celle des malveillances aussi.

La connaissance des attaques oriente la défense.

Sachez ce que les attaquants savent, découvrez vos vulnérabilités avant eux.

Commencez par des configurations sécurisées, par la gestion des droits, avant de penser à utiliser des produits ou logiciels complexes.

Utilisez des principes simples :

- Besoin d'en connaitre : ne donner les accès qu'aux personnes en ayant besoin, pour le temps nécessaire, et les retirer ensuite ;

- Moindre privilège : donner les privilèges minimums ;
- Séparation des rôles et des droits: séparation des taches critiques entre divers intervenants, afin d'éviter l'accumulation des privilèges (par exemple en implémentant des procédures de validation « 4 yeux ») ;
- Choisir entre les modèles : « Tout ce qui n'est pas explicitement interdit est autorisé » ou « Tout ce qui n'est pas explicitement autorisé est interdit ».

La prévention c'est l'idéal, la détection est nécessaire.

Partagez avec vos pairs. Ma détection est votre prévention.

Il faut être prêt à réagir. Et cela ne s'improvise pas.

Mise en œuvre de la sécurité

Le produit hérite de ses processus de production.

On ne peut pas se prémunir de toutes les menaces.

La sécurité ce sont des processus, jamais uniquement des logiciels ou des produits.

En sécurité il est essentiel de prioriser. Ne cherchez pas à atteindre le risque zéro. Concentrez-vous sur les mesures les plus efficaces.

Plus tard une vulnérabilité est découverte, plus la correction est coûteuse (en risque, en effort, en temps, en budget…).

Comportement

Formez-vous. Demandez des budgets formation, faites valoir vos droits à la formation. Partagez avec vos pairs, assistez à des conférences sécurité, à l'étranger de préférence. Faites-vous payer le voyage, l'hôtel. Partagez. Pas avec n'importe qui, dans des cercles de confiance et en faisant attention à la confidentialité. Mais partagez.

La sécurité a un coût en énergie, en temps, en argent, mais aussi en usages et en liberté.

Pensez ce que vous faites en termes éthiques, pour qui, pour quelle finalité ?

Avec de grands pouvoirs viennent de grandes responsabilités.

La fin ne justifie pas les moyens. Les moyens déterminent la fin.