

# Introduction à la Sécurité des Réseaux



Machine Enigma, *Imperial War Museum*, Londres

Introduction.....	4
Ordinateurs et sécurité des applications.....	4
Le réseau téléphonique (RTC).....	5
L'internet.....	6
L'internet sans fil, les réseaux radio.....	6
Transfert de fichiers et streaming.....	6
Roaming, VPN.....	7
Du Data Center au Cloud.....	7
L'émergence du monde connecté.....	8
Stuxnet, une arme logicielle.....	9
L'architecture logicielle selon REST.....	10
Principes de sécurité.....	11
Sécurité des Applications Distribuées.....	12
Sécurité et Réseau.....	12
Authentification - Autorisation.....	15
Identité et Organisations.....	16
Taxonomie des méthodes d'authentification.....	17
Mécanismes symétriques.....	17
Mécanismes Asymétriques.....	19
Les tunnels.....	19
Classification des types d'attaques.....	19
Les faiblesses du protocole IP.....	20
Les solutions sécurisées classiques.....	21
IPSEC.....	21
TLS/SSL.....	23
Au sujet du Phishing.....	25
Kerberos.....	25
EAP.....	26
Limitation des protections offertes par les pare-feu.....	27
De la faiblesse des systèmes informatiques.....	28
Faites moi confiance ? 20 ans de bugs et heuristiques.....	28
Quelques paradigmes de défense.....	32
Exemple de classification des attaques pour les systèmes embarqués.....	33
Au sujet des normes Critères Communs (CC).....	33
L'intrusion.....	35
Programmes malveillants, virus, vers.....	35
Exemple le ver blaster.....	36
Au sujet des botnets.....	36
Au sujet des rootkits.....	37
Quelques techniques malveillantes classiques.....	37
Le buffer overflow.....	37
Le fuzzing.....	38
L'injection SQL.....	38

Le Cross Site Scripting .....	38
Un exemple de stratégie de défense.....	39
Intégrité du code, obfuscation, TPM.....	39
Les Canaux Cachés.....	41
Single Power Analysis .....	41
Differential Power Analysis.....	41
Récents attaques sur les fonctions de hash MD5 et SHA-1 .....	43
Quelques TOP10 d'attaques .....	43
Le TOP9 des menaces visant le cloud computing en 2013 .....	43
OWASP TOP10 2013 .....	45
Principes de sécurité du système Android.....	47
Attaques TLS diverses .....	48
L'attaque par renégociation (2009).....	48
L'attaque BEAST (2011).....	48
L'attaque Lucky Thirteen (2013).....	50
Attaque TLS Triple handshake .....	52
HeartBleed 2014 .....	52
L'attaque RC4 du L'attaque Royal Holloway (2013).....	53

# Introduction à la Sécurité des Réseaux 2014-2015

---



## Introduction

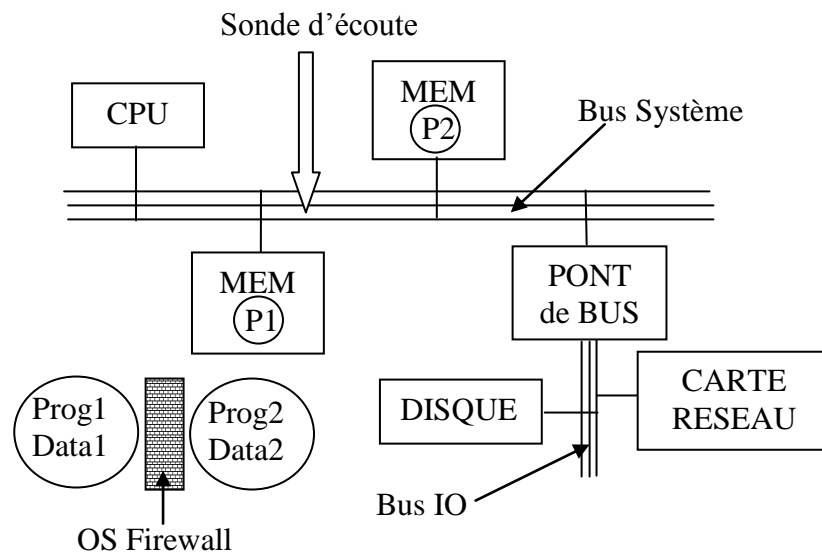
Une application distribuée est réalisée par un ensemble d'entités logicielles logiquement autonomes, qui produisent, consomment et échangent des informations ( $OUT_i = PROG(IN_i)$ ).



## *Ordinateurs et sécurité des applications*

Dans un premier temps les composants logiciels des applications étaient logés dans un même système informatique, constituant de fait leur média de communication (parfois dénommé *gluware*). Le bus système permet le transfert des informations stockées en mémoire, les modules logiciels sont réalisés par des processus gérés par le système d'exploitation. La sécurité est uniquement dépendante des caractéristiques du système d'exploitation, par

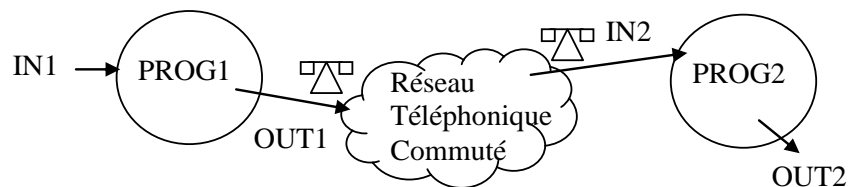
exemple en termes de gestion des droits utilisateurs, ou d'isolement des processus.



### Le réseau téléphonique (RTC)

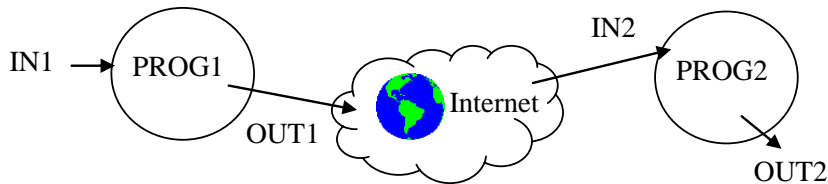


Dans une deuxième période l'application distribuée est répartie entre plusieurs systèmes informatiques reliés entre eux par des liens de communications supposés sûrs (c'est à dire qu'il est difficile d'enregistrer ou de modifier l'information transmise) tels que modems ou liaisons spécialisées (X25, RNIS ...). Nous remarquerons à ce propos qu'il est possible de sécuriser une liaison de type point à point par un dispositif matériel de chiffrement.



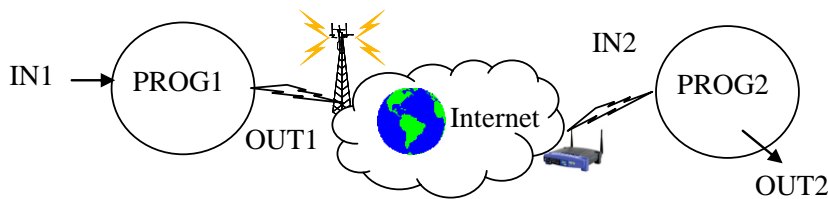
### *L'internet*

Enfin l'émergence de la toile d'araignée mondiale a permis de concevoir des systèmes distribués à l'échelle planétaire, les composants logiciels sont répartis sur des systèmes informatiques hétéroclites, le réseau n'est pas sûr, le nombre d'utilisateurs est important. La sécurité devient un paramètre critique et tente de concilier des contraintes à priori antinomiques telles que, nécessité économique d'utiliser Internet, et impérative résistance à la piraterie informatique ou à l'espionnage.



### *L'internet sans fil, les réseaux radio*

La dernière révolution des communications s'appuie sur les technologies de réseaux IP sans fil, tels que Wi-Fi ou WiMAX. Les liens filaires symboles d'une connectivité volontaire et contrôlée s'estompent; l'infrastructure du réseau devient diffuse et invisible. Un nouveau besoin de sécurité s'affirme, le contrôle des accès réseaux.



### *Transfert de fichiers et streaming*

Les réseaux de communication transportent les données produites et consommées par les systèmes distribués. Ils offrent deux types de services fondamentaux, le *transfert de fichiers* (un ensemble de données prédéfinies) et la diffusion d'information, variant dans le temps, *en mode flux*. La première catégorie comporte des services tels que le courrier électronique (POP,

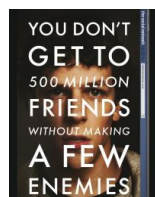
SMTP...), le WEB (HTTP) et diverses méthodes d'échange d'information (FTP, NNTP, ...). La deuxième catégorie regroupe les protocoles relatifs au multimédia ou à la téléphonie sur IP, par exemple RTP, H323, ou SIP.

### ***Roaming, VPN***

Les fournisseurs de services internet gèrent un ensemble de serveurs qui stockent les données et les applications de leurs clients (base de données, messageries, fichiers...). Le *roaming* consiste à autoriser l'accès à distance de ces ressources, ce qui implique également la gestion sécurisée de la mobilité des utilisateurs, typiquement à l'aide de protocoles VPN (*Virtual Private Network*).

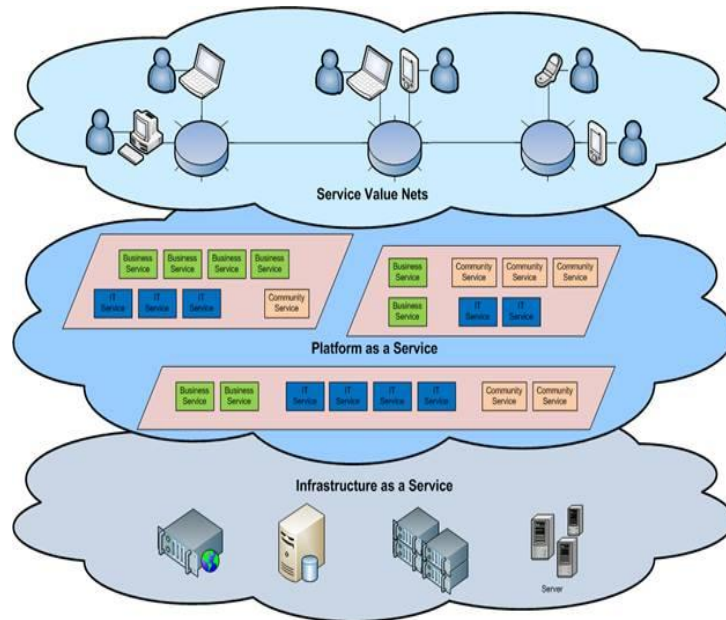
### ***Du Data Center au Cloud***

Durant la période 2000-2010 les grands fournisseurs de services de l'Internet (tels que Google, facebook) ont développé une technologie s'appuyant sur des *Data Centers*, c'est-à-dire des sites informatiques regroupant jusqu'à un million d'ordinateurs (cartes mère), pour un coût de l'ordre de 600 M\$, et consommant environ 100 MW (le dixième d'un réacteur nucléaire). Cette approche est à la base du concept du *Cloud Computing*, le client est équipé d'un terminal léger comportant un navigateur ou une machine virtuelle, la sécurité d'accès au service repose sur des technologies WEB.



Une plateforme de cloud computing est divisée en trois couches fonctionnelles:

- Le SaaS ou *Software as a Service*. Par exemple Salesforce.com met à la disposition de ses clients un ensemble de logiciels dédiés à la gestion des entreprises.
- PaaS ou *Platform as a Service*. C'est un environnement (APIs, Machine Virtuelles) permettant de déployer des applications par exemple Google App Engine ou Microsoft Azure.
- IaaS, ou *Infrastructure as a Service*. C'est typiquement un service de location de ressources telles que serveurs ou stockage de données, par exemple Amazon EC2. La propriété essentielle est la notion d'élasticité, c'est-à-dire l'adaptation dynamique des ressources à la demande.



### *L'émergence du monde connecté*

En 2013 plus d'un milliard de smartphones ont été vendus; ce chiffre marque le basculement vers le paradigme du *monde connecté*. Par exemple la maison connectée (*smart home*), la voiture connectée (*connected car*), les objets connectés (lunette, montre, vêtements...), les capteurs médicaux (*body area network*, BAN), les immeubles intelligents (*smart building*), la ville intelligente (*smart city*), et de manière générale l'internet des choses (*Internet of Things IoT*).

Selon un white paper<sup>1</sup> de cisco le nombre d'objets connectés était de 10 milliards en 2010 et pourrait atteindre 50 milliards en 2020.

Le moteur de recherches SHODAN (<http://www.shodanhq.com>) recense les dispositifs connectés. De nombreuses attaques/intrusions s'appuient sur l'usage de mots de passe par défaut.

La RFC 7228 ("*Terminology for Constrained-Node Networks*", 2014) propose un classification en 3 classes des nœuds de l'IoT à ressources réduites

<sup>1</sup> "The Internet of Things. How the Next Evolution of the Internet Is Changing Everything"  
[http://www.cisco.com/web/about/ac79/docs/innov/IoT\\_IBSG\\_0411FINAL.pdf](http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf)



- Classe C0, RAM  $\ll$  10KB, code  $\ll$  100KB, ces systèmes (capteurs) n'ont pas les ressources nécessaires pour gérer des connexions sécurisées

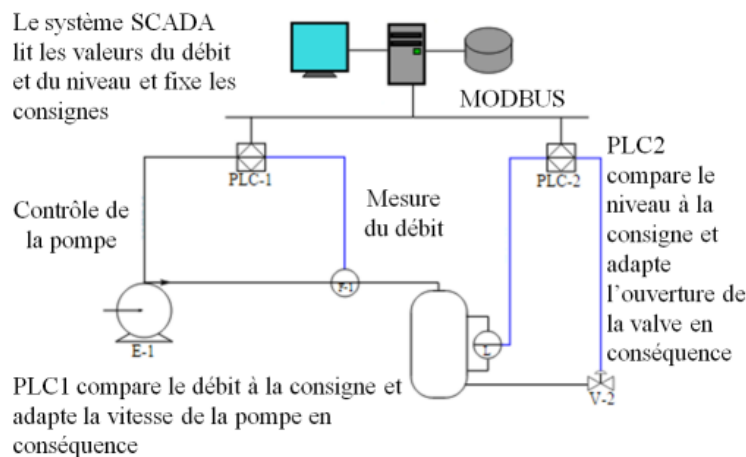
- Classe C1, RAM #10KB, code #100KB, les ressources systèmes sont suffisantes pour assurer des connexions (TC-UDP/IP) sécurisées selon des protocoles peu complexes.

- Classe C2, RAM #50KB, code #500KB, les ressources systèmes autorisent des connexions à des dispositifs TCP/IP (notebook, tablettes) classiques.

Le protocole COAP (RFC 7252) sécurise les échanges d'information de capteurs en utilisant DTLS (TLS avec un transport UDP)

L'industrie, en particulier le contrôle des processus industriels (usines...) utilise également des technologies connectées. Par exemple le système de contrôle et d'acquisition de données SCADA<sup>2</sup> (*Supervisory Control And Data Acquisition*), comporte des dispositifs de contrôle tels que PLC (*programmable logic controllers*) ou RTU (*remote terminal unit*) communiquant via le protocole MODBUS fonctionnant en mode Master/Slave (sur des liaisons séries RS-232, RS-485) ou Client/Server (TCP/IP).

### Stuxnet, une arme logicielle



Le ver informatique *Stuxnet* découvert en 2010 a permis la destruction de machines industrielles (un millier de centrifugeuses...) en attaquant la

<sup>2</sup> [http://www.schneider-electric.com/solutions/ww/fr/med/20340568/application/pdf/1485\\_se-whitepaper-letter-scadaoverview-v005.pdf](http://www.schneider-electric.com/solutions/ww/fr/med/20340568/application/pdf/1485_se-whitepaper-letter-scadaoverview-v005.pdf)

plateforme SCADA WinCC/PCS7 de Siemens et en reprogrammant des PLCs (automates S7), depuis des machines *Windows* préalablement corrompues (des serveurs winCC).

### **L'architecture logicielle selon REST**

Dans sa thèse de doctorat, "*Architectural Styles and the Design of Network-based Software Architectures*", publiée en 2000 Thomas Fielding (Université d'Irvine, CA, USA) définit le concept d'architecture REST (Representational State Transfer).

Il introduit la notion d'architecture logicielle (distribuée) sous forme d'un assemblage de trois types d'éléments, les composants, les connecteurs et les données.

Un composant (*component*) est un bloc abstrait d'instructions et d'états machine qui traite des données reçues via son interface.

Un connecteur (*connector*) est un mécanisme abstrait qui organise la communication, la coordination, et la coopération entre composants.

Un *datum* est un élément d'information reçu ou transmis par un composant via un connecteur.

Une architecture REST est une généralisation du WEB; ses principales caractéristiques sont les suivantes

- Deux types d'entités les clients et les serveurs (*Client-Server*).
- Pas d'états machine (*stateless*). Les requêtes échangées sont indépendantes les unes des autres.
- Usage de caches (*cache*) pour meilleure efficacité du système.
- Interface Uniforme (*uniform interface*) en particulier pour l'identification et l'usage des ressources.
- Système structuré en couche (*layered system*), mais présentant des interfaces uniques.
- Code à la demande (*on demand code*) exécuté par le client.

Dans un contexte REST la sécurité du réseau consiste à garantir l'intégrité et la confidentialité des données échangées entre composants.

## Principes de sécurité.

Classiquement la sécurité s'appuie sur cinq concepts de base.

**L'identification** (*identity*). L'utilisateur d'un système ou de ressources diverses possède une identité (une sorte de clé primaire d'une base de données) qui détermine ses lettres de crédits (*credential*) et ses autorisations d'usage. Cette dernière peut être déclinée de multiples manières, compte utilisateur (login) d'un système d'exploitation ou techniques biométriques empreinte digitale, empreinte vocale, schéma rétinien...

**L'authentification** (*authentication*). Cette opération consiste à faire la preuve de son identité. Par exemple on peut utiliser un mot de passe, ou une méthode de défi basée sur une fonction cryptographique et un secret partagé. L'authentification est **simple** ou **mutuelle** selon les contraintes de l'environnement.

**La confidentialité** (*privacy*). C'est la garantie que les données échangées ne sont compréhensibles que pour les deux entités qui partagent un même secret souvent appelé *association de sécurité* (SA). Cette propriété implique la mise en œuvre d'algorithmes de chiffrements soit en mode flux (octet par octet, comme par exemple dans RC4) soit en mode bloc (par exemple une série de 8 octets dans le cas du DES).

**L'intégrité** des données (MAC, Message Authentification). Le chiffrement évite les écoutes indiscretes, mais il ne protège pas contre la modification des informations par un intervenant mal intentionné. Des fonctions à sens unique (encore dénommées empreintes) telles que MD5 (16 octets) ou SHA1 (20 octets) réalisent ce service. Le MAC peut être associé à une clé secrète, telle la procédure HMAC(Clé, Message), *Keyed-Hashing for Message Authentification*.

**La non-répudiation**. Elle consiste à prouver l'origine des données. Généralement cette opération utilise une signature asymétrique en chiffrant l'empreinte du message avec la clé RSA privée de son auteur (RSA(empreinte(Message))).

On cite fréquemment un sixième attribut relatif à aux notions de **sûreté de fonctionnement, disponibilité, et résilience** du système.

Remarquons également que la sécurité implique le partage de confiance entre les différents acteurs de la chaîne. Pour partager un secret il faut avoir confiance dans les capacités des parties concernées à ne pas le divulguer. Ainsi les infrastructures à clés publiques (PKI) supposent que l'on fasse

confiance aux entités qui produisent les clés privées, et les signatures des certificats.

La confiance est une relation sans propriétés particulières.

**Réflexivité**, ai-je confiance en moi-même (pas dans tous domaines).

**Symétrie**, je fais confiance au pilote de l'avion ou au chirurgien, la réciproque n'est pas forcément vraie.

**Transitivité**, j'ai confiance dans le président, le président a confiance en la présidente, je n'ai pas obligatoirement confiance dans la présidente.

Les infrastructures PKI supposent une transitivité de la relation de confiance. Le client du réseau et un serveur d'authentification partagent une même autorité de certification (CA), qui crée une classe de confiance basée sur une relation R (R signifiant= «fait confiance à»).

(Client R CA) ET (Serveur R CA) => (Client R Serveur)

### Sécurité des Applications Distribuées

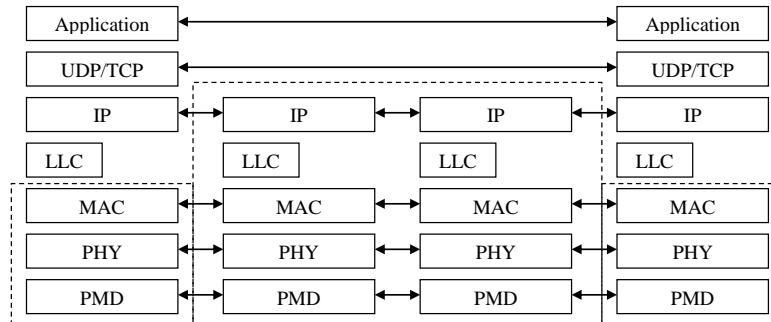
La sécurité globale s'appuie sur le triptyque:

- 1) la sécurité des applications clientes et serveurs,
- 2) la sécurité des plateformes informatiques qui exécutent ces applications (Operating System, Hardware),
- 3) la sécurité des protocoles de communication et du réseau qui transporte ces éléments.

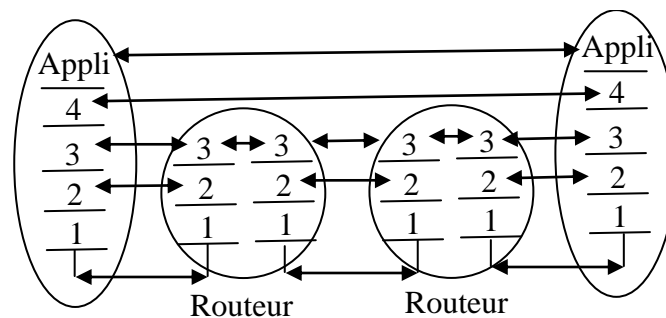
### Sécurité et Réseau

Un réseau assure le transport des messages échangés entre deux applications distantes. Dans le modèle OSI les services déployés par le réseau sont classés en sept couches, physique, données, réseau, transport, session, présentation et application. Le modèle classique des réseaux TCP/IP ne comporte que 5 couches, physique (PMD+PHY), données (MAC+LLC), réseau (IP), transport (UDP+TCP) et applications.

Dans cette section nous ne prendrons en compte que ce dernier modèle, qui est aujourd'hui le standard *de facto* pour l'échange d'informations numériques.



Des mécanismes tels que confidentialité ou intégrité des données peuvent être intégrés à tous les niveaux et sur les différents tronçons (arcs) qui composent le réseau. La gestion des clés cryptographiques sera par exemple réalisée manuellement. L'identification l'authentification la non répudiation les autorisations sont des procédures mises en œuvre dans le réseau d'accès (sans fil par exemple), le réseau de transport (IP), le réseau de destination (intranet ...). De même ces services peuvent également être offerts au niveau applicatif.



Schématiquement nous classerons les infrastructures de sécurité des réseaux en cinq catégories,

A- Le chiffrement au niveau physique sur des liaisons point à point. Par exemple cryptographie quantique (PMD), saut de fréquences pseudo aléatoire, ou chiffrement 3xDES du flux octets (une méthode couramment déployée par les banques). Dans ces différentes procédures les clés sont distribuées manuellement.

B- Confidentialité, intégrité de données, signature de trames MAC. C'est la technique choisie par les réseaux sans fil 802.11. La distribution des clés est réalisée dans un plan particulier (décrit par la norme IEEE 802.1x). Dans

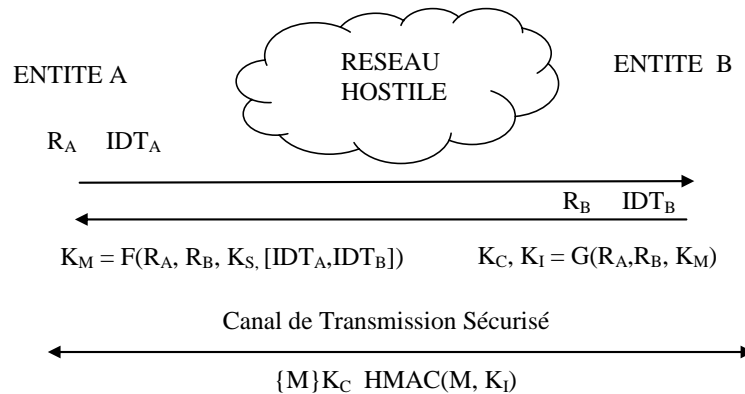
ce cas on introduit la notion de contrôle d'accès au réseau LAN, c'est à dire à la porte de communication avec la toile d'araignée mondiale. C'est une notion juridique importante, le but est d'interdire le transport des informations à des individus non authentifiés (et donc potentiellement malveillants...)

C- Confidentialité, intégrité de données, signature des paquets IP et/ou TCP. C'est typiquement la technologie IPSEC en mode tunnel. Un paquet IP chiffré et signé est encapsulé dans un paquet IP non protégé. En effet le routage à travers l'Internet implique l'analyse de l'entête IP, par les passerelles traversées. IPSEC crée un tunnel sécurisé entre le réseau d'accès et le domaine du fournisseur de service. On peut déployer une gestion manuelle des clés ou des protocoles de distribution automatisés tels que ISAKMP et IKE. La philosophie de ce protocole s'appuie sur la libre utilisation du réseau d'accès ce qui n'est pas sans soulever des problèmes juridiques. Par exemple des pirates protègent leurs échanges de données, il est impossible aux réseaux traversés de détecter leur complicité dans le transport d'informations illégales.

D- Insertion d'une couche de sécurité additive assurant la protection d'application telles que navigateurs WEB ou messageries électroniques. Par exemple le protocole SSL basé sur la cryptographie asymétrique réalise cette fonction. Généralement ce dernier conduit une simple authentification entre serveur et client. Il utilise un secret partagé (Master Secret) à partir duquel on dérive des clés de chiffrements utilisées par l'algorithme négocié entre les deux parties. Par exemple dans le cas d'une session entre un navigateur et un serveur bancaire, le client authentifie son service bancaire. Une fois le tunnel sécurisé établi le client s'authentifie à l'aide d'un login et d'un mot de passe. Il obtient alors une identité temporaire associée à un simple cookie.

E- Gestion de la sécurité par l'application elle-même. Ainsi le protocole S-MIME réalise la confidentialité, l'intégrité et la signature des contenus critiques d'un message électronique.

### Authentification - Autorisation



La procédure d'authentification d'une paire d'entités informatiques, parfois dénommée phase d'autorisation, consiste typiquement à échanger les identités ( $IDT_A$  et  $IDT_B$ ) d'un couple d'interlocuteurs (appelés client/serveur ou initiateur/répondeur), deux nombres aléatoires ( $R_A$ ,  $R_B$ ) formant un identifiant unique de la session ( $RA || RB$ ), puis d'effectuer un calcul cryptographique ( $F$ , une fonction de type Pseudo Random Function - PRF).

Ce dernier produit, à l'aide d'une valeur secrète ( $K_S$ ) un secret maître ( $K_M$ ), à partir duquel on déduit (à l'aide d'une fonction  $G$ , de type Key Data Function - KDF) des clés de chiffrement ( $K_C$ ) et d'intégrité ( $K_I$ ) permettant de créer un canal sécurisé.

Dans un contexte de cryptographie symétrique la clé  $K_S$  est *statique* et distribuée manuellement; dans un contexte de cryptographie asymétrique la clé  $K_S$  sera par exemple générée par  $A$ , mais chiffrée par la clé publique ( $e, n$ ) de  $B$  ( $K_S^e \bmod n$ ), ou bien déduit d'un échange de Diffie-Hellman ( $K_S = g^{xy}$ )

La protection de l'identité est une préoccupation croissante avec l'émergence des technologies sans fil. Il existe divers mécanismes permettant d'obtenir cette propriété avec des degrés de confiance différents, par exemple grâce à la mise en œuvre de pseudonymes (tel que le TIMSI du GSM), du protocole de Diffie-Hellman, ou du chiffrement de certificats à l'aide la clé publique du serveur.

## Identité et Organisations

L'identité détermine les autorisations et/ou la localisation d'une personne, d'un objet (parfois intelligent), ou des deux, relativement à une organisation (état, privée...). L'authentification est la procédure qui consiste à faire la preuve d'une identité. On peut utiliser divers moyens,

- Ce qui je suis, méthodes biométriques, éventuellement multi modales (passeport électronique...)
- Ce que je connais, mots de passe, ....
- Ce que je possède, cartes à puce, jeton...

Voici quelques exemples d'organisations gérant des d'identités

- **Les états** (identité des citoyens et des visiteurs). Les identités utilisent par exemple la biométrie (programme *US-Visit*), les passeports électroniques, les cartes d'identité munies de puces.

- **Le WEB** (Applications WEB). Il existe de multiples infrastructures d'identités telles que, *Single Sign On (SSO)*, Microsoft Passport, Liberty Alliance, OPENID...

- **L'industrie** (localisation, inventaire...). L'identité des objets est associée à des étiquettes (code barre, code 2D) ou des RFIDs.

- **Les services informatiques** gèrent des ordinateurs personnels. L'identité d'une machine est par exemple assurée par un *Trusted Platform Module (TPM)*.

- **Les réseaux bancaires** réalisent des opérations de paiement associées à des cartes BO' ou EMV.

- **Les réseaux de communication** réalisent des échanges de données, et mettent en œuvre différentes identités : couple login mot de passe, carte SIM ou USIM , Wi-Fi et EAP-ID, jetons divers (tels que RSA SecurID)



## Taxonomie des méthodes d'authentification

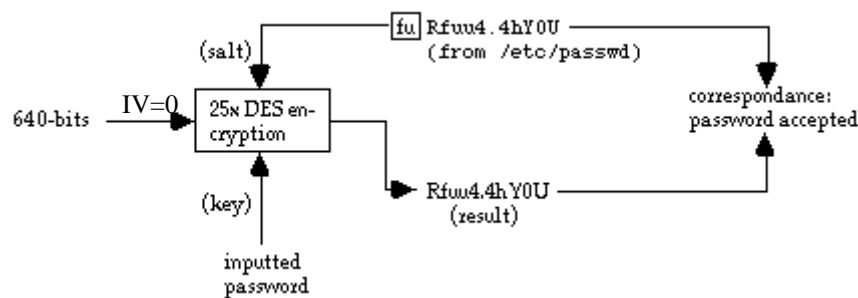
Nous classons les méthodes d'authentification en trois catégories, symétriques (secrets partagés), asymétriques (basées sur RSA en règle générale) et tunnels.

### Mécanismes symétriques

Nous divisons les méthodes symétriques en trois classes

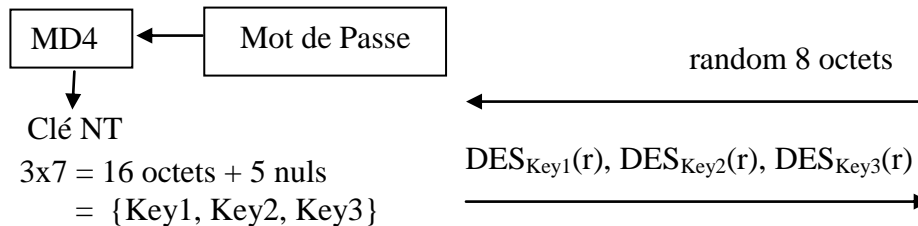
- **Les mots de passe.** Un mot de passe est une suite de caractères qui peut être facilement mémorisée par un être humain. Il ne s'agit pas d'un nombre aléatoire mais au contraire d'une concaténation de mots, de chiffres et de signes. De tels secrets peuvent être devinés à l'aide d'une attaque par dictionnaire, c'est à dire un programme utilisant une base de données pour la génération de ces valeurs. Ainsi les méthodes EAP-MSCHAPv2 ou LEAP reposent sur la clé NT, c'est à dire l'empreinte MD4 (soit 16 octets) d'un mot de passe.

Dans les anciens systèmes UNIX la liste des empreintes des mots de passe est stockée dans le fichier `/etc/passwd`. Le fichier `/etc/shadow` lisible uniquement en mode `root` est aujourd'hui préféré. Un mot de passe est par exemple une chaîne d'au plus 8 caractères ASCII, convertie en une clé DES de 56 bits ( $8 \times 7$  bits). La fonction `crypt(key, salt)` réalise dans ce cas 25 chiffrements DES consécutifs (avec une valeur initiale  $IV=0$ ), dont chacun comporte une permutation choisie parmi 4096 possible (soit 12 bits, c'est le paramètre *salt*, deux caractères choisis parmi 64 valeurs possibles *a-z A-Z 0-9 . /*). D'autres algorithmes de génération d'empreinte sont supportés, tels que la fonction MD5.



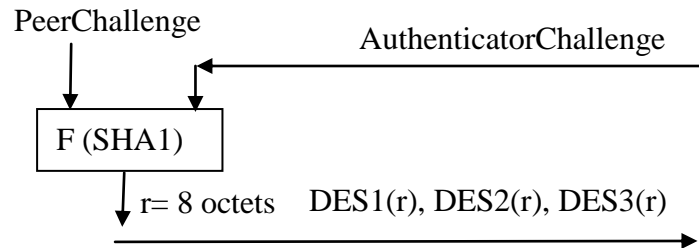
Dans l'univers Microsoft la sécurité d'un ordinateur personnel est fortement corrélée au mot de passe de son utilisateur. Ce dernier n'est jamais stocké en clair dans la mémoire de la machine. A partir d'un mot de passe

on calcule une empreinte MD4 de 16 octets, mémorisée par le système hôte. Cette valeur, parfois nommée clé NT ou *NtPasswordHash* est complétée par cinq octets nuls. On obtient ainsi 21 octets interprétés comme une suite de trois clés DES (de 56 bits chacune).



La méthode MSCHAPv1 est une authentification simple, le serveur d'authentification produit un nombre aléatoire de 8 octets, l'authentifié utilise ses trois clés DES pour chiffrer cet aléa, ce qui génère une réponse de 24 octets.

MSCHAPv2 est une extension du protocole précédent, le serveur d'authentification délivre un nombre aléatoire de 16 octets (*AuthenticatorChallenge*), le client calcule un nombre 8 octets à partir de cette valeur, d'un aléa (*PeerChallenge*) qu'il génère et du nom de l'utilisateur (login). Ce paramètre est chiffré de manière analogue à MSCHAPv1 par la clé NT et l'on obtient une valeur de 21 octets. Dans une plateforme Microsoft un annuaire stocke le nom des utilisateurs et leur clé NT.



- **Les secrets partagés** (PSK, Pre-Shared-Key). Il s'agit en fait d'un nombre aléatoire, dont la taille est l'ordre de 128 à 160 bits. Un cerveau humain éprouve beaucoup de difficultés à mémoriser ces informations. Le stockage sécurisé du secret est réalisé par exemple par le système d'exploitation d'un ordinateur personnel ou par une carte à puce.

- **Le mode provisioning.** Dans les réseaux GSM ou UMTS une base de donnée centralisée (*Host Location Register*) gère les comptes utilisateurs, en particulier leur PSK. Afin d'éviter des interrogations fréquentes les

méthodes d'authentification (A3/A8, Milenage) sont conçues de telle manière que le site central puisse produire des vecteurs d'authentification (triplets GSM ou quadruplets UMTS), réutilisables par des agents de confiance (tels que les *Visitor Location Register* par exemple).

### **Mécanismes Asymétriques**

Ces procédures sont basées sur des algorithmes tels que ECC, RSA ou Diffie-Hellman. Le protocole SSL/TLS est généralement utilisé pour le transport de ces échanges.

### **Les tunnels**

Ainsi que nous l'avons souligné précédemment les méthodes d'authentification basées sur des mots de passe, sont sujettes à des attaques par dictionnaire. Ainsi le protocole MSCHAP assure une protection jugée raisonnable dans des environnements sûrs (par exemple des intranets ou des connexions par modem), mais n'est plus adapté lors d'une mise en œuvre dans un milieu hostile, tel que IP sans fil, abritant potentiellement de nombreuses oreilles électroniques indiscrettes.

Les tunnels, s'appuyant fréquemment sur la technologie SSL/TLS, protègent un dialogue d'authentification grâce au chiffrement des données échangées. Il existe aujourd'hui de multiples standards devenus nécessaires, en raison des nombreux logiciels disponibles sur le WEB qui cassent les protocoles à base de mot de passe.

### **Classification des types d'attaques**

On peut classer<sup>3</sup> les attaquants d'un système en trois catégories



- **Classe I - (*clever outsiders*)**: L'attaque accidentelle. Un utilisateur constate de manière fortuite un défaut du système.

- **Classe II - (*knowledgeable insiders*)**. L'attaque individuelle ou par de petites communautés, mais avec des moyens limités. L'attaquant réalise un investissement modeste, mais espère un gain financier ou de notoriété.

<sup>3</sup> DG Abraham, GM Dolan, GP Double, JV Stevens, "Transaction Security System", in *IBM Systems Journal* v 30 no 2 (1991) pp 206 229

- **Classe III - (*funded organisations*)**. L'attaque par des organisations (Etats, ...) disposant de moyens importants. La logique financière n'est pas forcément un but.

### Les faiblesses du protocole IP

Par nature un réseau est sensible au déni de service, au niveau physique (brouillage divers...) ou logique (destruction/modification des paquets ou des trames).

Le protocole ARP (*Address Resolution Protocol*) réalise une correspondance entre une adresse MAC et une adresse IP. Dans l'attaque dite *ARP spoofing* l'attaquant forge une trame de réponse (*ARP.response*) erronée. Il en résulte un détournement du trafic IP.

Un paquet IP comporte typiquement un en tête de 20 octets démunis d'attributs cryptographiques de sécurité. La confidentialité et l'intégrité des données transportées ne sont donc pas assurées.

Le mécanisme de segmentation est difficilement analysable par les pare-feu. En effet, seul le premier segment IP contient l'entête du protocole supérieur transporté, par exemple TCP ou UDP. De même le réassemblage peut entraîner un problème de déni de service pour la machine de réception (la taille maximale d'un paquet IP est de 65535 octets et le temps de réception des fragments est indéterminé).

La correspondance adresse IP nom de machine est typiquement réalisée par le protocole DNS qui n'offre aucun service d'authentification ou d'intégrité.

Lorsqu'un pare-feu autorise les paquets ICMP, il s'expose à de possibles canaux cachés, utilisés par exemple pour dialoguer avec des chevaux de Troie, dont les informations sont transportées par des paquets ICMP.response.

Le protocole TCP ne propose aucune authentification du serveur lors de l'ouverture d'une session (paquets SYN et ACK+SYN).

Le SYN-Flooding est une technique d'attaque de déni de service qui consiste à générer en grand nombre de paquets TCP-SYN.

L'analyse des ports TCP (en mode serveur) ouverts d'un nœud IP (port scan) repose sur la possibilité de forger librement des paquets TCP-SYN.

Il est possible de mettre fin à une session TCP à l'aide de paquets TCP-RESET. Connaissant l'adresse IP du client en supposant une fenêtre de

réception (RWIN) de  $2^{14}$  et une plage de ports éphémères de  $2^{10}$  le nombre de paquets nécessaire est de l'ordre de  $2^{32}/2^{14}*2^{10} = 2^{28}$ .

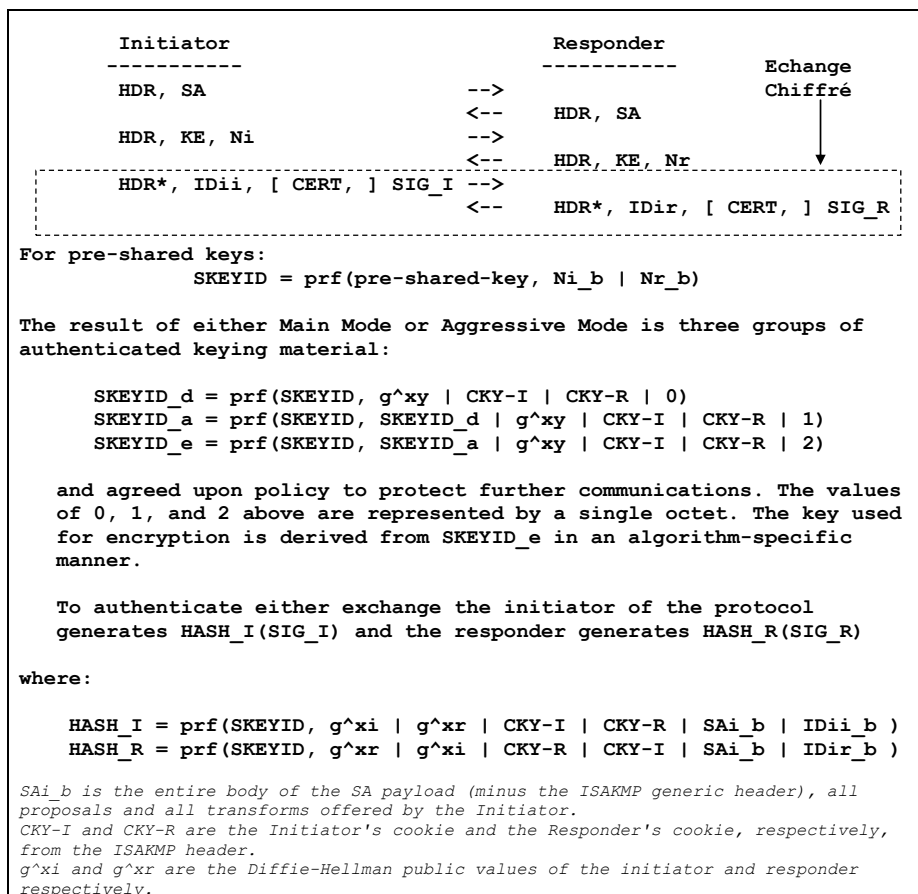
De nombreux protocoles (PPP, POP, FTP,...) mettent en œuvre des mots de passe transmis en clair sur le réseau. Certaines messageries ne supportent pas un transport sécurisé par SSL/TLS.

Globalement le modèle Internet présente une insécurité quasi globale, notamment au niveau de la messagerie...., et pourtant il fonctionne quotidiennement.

### **Les solutions sécurisées classiques**

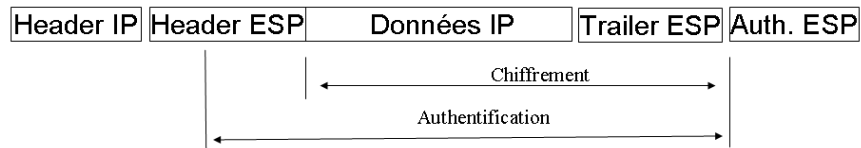
#### ***IPSEC***

IPSEC assure l'intégrité, l'authentification et la confidentialité des charges transportées par le protocole IP. Les ressources cryptographiques sont déduite d'une **association de sécurité** (SA) basée sur un secret partagé entre les deux entités de la session sécurisée IP. La difficulté de déploiement du protocole IPSEC est liée au mécanisme d'établissement d'un SA, à l'aide du protocole IKE (qui réalise un premier tunnel sécurisé à l'aide d'un protocole de Diffie-Hellman). Bien que l'usage de certificats soit possible, c'est généralement une clé pré définie (*PreShareKey*) qui permet l'authentification des deux extrémités de la communication.

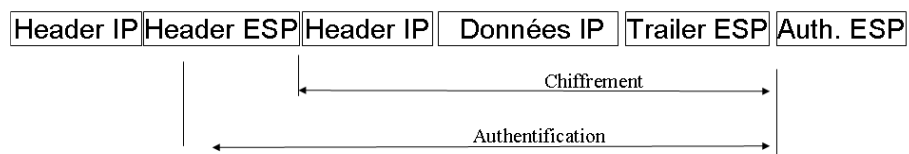


Protection d'identité et calcul de clé dans le protocole IKE, en phase 1,  
*presared key main mode.*

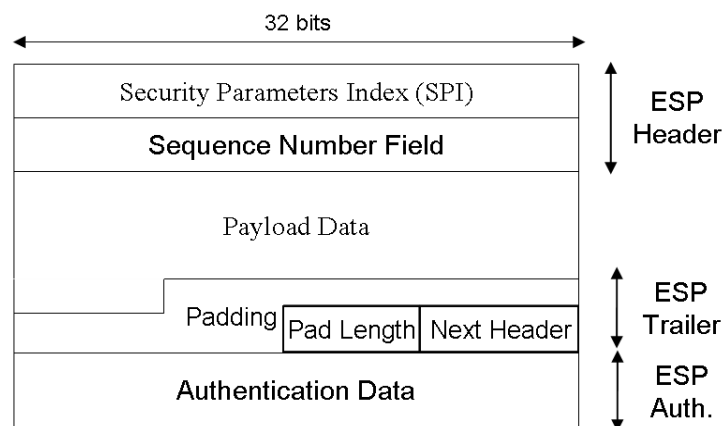
### Mode transport



### Mode tunnel



### Mode Transport et Tunnel dans le protocole ESP, IPSEC

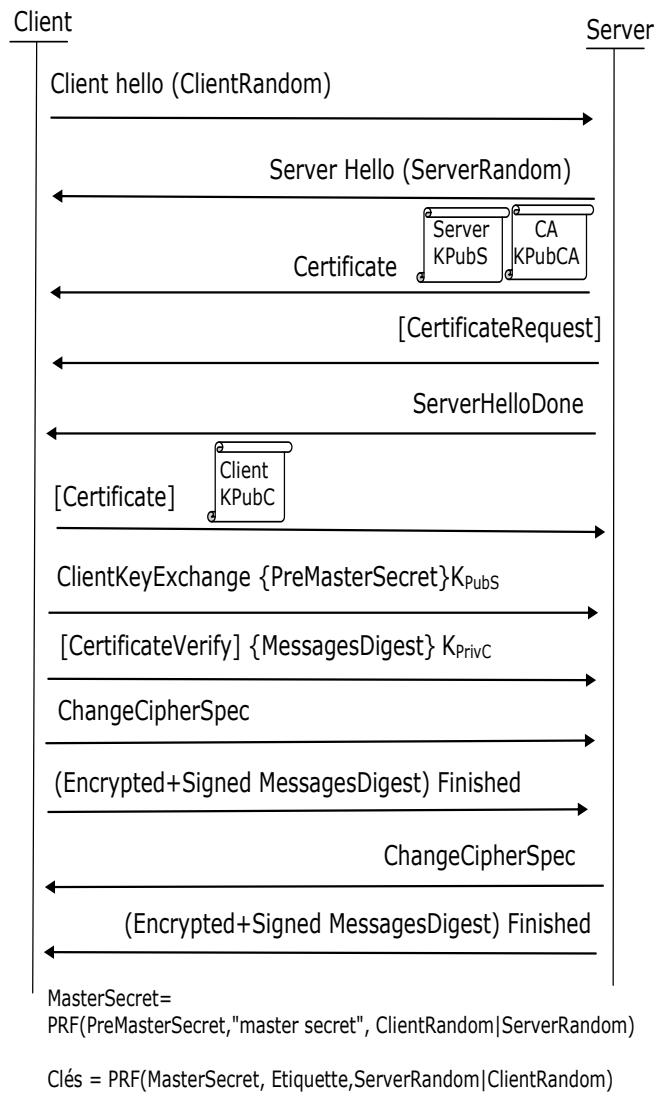


En tête du protocole ESP.

### TLS/SSL

Le protocole SSL/TLS délivre des services analogues à IPSEC, mais au niveau applicatif. Cependant le mécanisme d'établissement du secret partagé maître repose généralement sur des clés RSA et des certificats X509. De manière classique le certificat du serveur est vérifié relativement à une chaîne de confiance (suite de certificats délivrés à partir d'un certificat racine). Côté client les points critiques sont l'installation d'un nouveau

certificat d'une autorité de certification, et la vérification des certificats serveurs.



### Ouverture d'une session TLS



### *Au sujet du Phishing*

L'apparition des attaques dites de *Phishing*, dont le but est de collecter des numéros de carte bancaire ou des informations permettant d'accéder à des comptes bancaires résulte de deux faiblesses clés de l'Internet:

- 1) La difficulté à authentifier au niveau TCP/IP les serveurs distants (DNS non sécurisé, paquet IP non authentifié, détournement de session TCP...), et
- 2) La difficulté de vérification des certificats serveurs.

### *Kerberos*

Kerberos est un protocole développé par le MIT dont les deux versions majeures sont v4 et v5. La version 4 s'appuie sur l'algorithme cryptographique DES; La version 5 supporte 3xDES et AES. C'est un standard IETF, RFC 1510 (kerberos v5, 1993).

Kerberos utilise un paradigme à base de tickets établissant une preuve d'identité entre deux entités (un utilisateur et un service).

Le système comporte trois classes de composants, le client, le KDC (*Key Distribution Center*), et des serveurs d'applications. Le KDC regroupe un serveur d'authentification (AS), un serveur de tickets (TGS, *Ticket Granting Service*) et une base de données clients. Les messages sont codés selon la syntaxe ASN.1

Le *Realm* est un nom de domaine lié (example.com) à une autorité d'authentification. Un utilisateur appartient à un domaine lorsqu'il partage un mot de passe ou une clé cryptographique avec ce dernier.

Le *Principal* est la clé d'identification dans la base de données client.

Le *Ticket* est une donnée délivrée par le serveur d'authentification qui comprend:

- Une zone d'information chiffrée avec la *clé de l'utilisateur*, comportant une date de validité du ticket et une *clé de session du service*.
- Une zone d'information chiffrée avec la *clé du service*, incluant une date de validité et une *clé de session du service*;

La base de données du KDC stocke toutes les informations associées à un principal (mot de passe, clé, etc...).

Le serveur d'authentification réalise l'authentification d'un utilisateur à l'aide de son mot de passe; il délivre un ticket d'authentification, *Ticket Granting Ticket*, ou TGT, identifié par un principal. Le *Ticket Granting Server* (TGS) génère des tickets de service pour un utilisateur authentifié, c'est-à-dire muni d'un TGT.

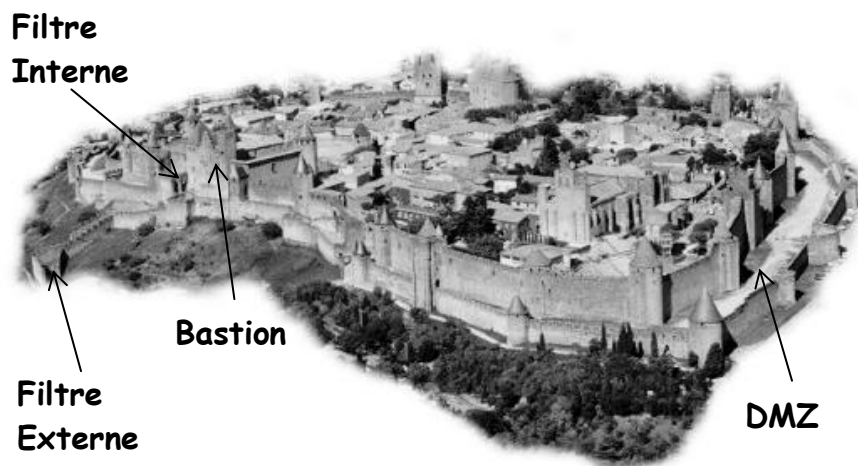
La clé de session (*Session Key*) est dans le cas d'un utilisateur du KDC une clé cryptographique (DES, 3xDES, AES) déduite de son mot de passe; et dans le cas d'un service elle est générée par le KDC.

Un *authenticator* est une structure d'authentification comportant l'identité de l'utilisateur et une date chiffrée avec la clé de session service. Un *authenticator* lié à un ticket de service évite la duplication illicite de ticket. La présence d'un *authenticator* «frais» prouve la connaissance de la clé de session de service.

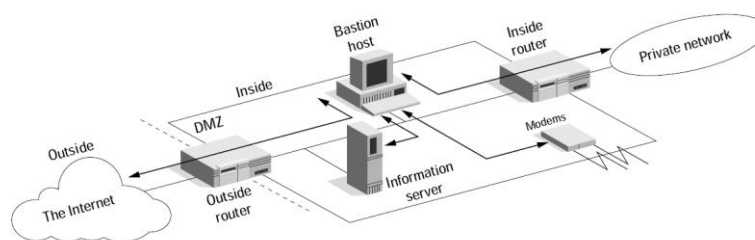
### **EAP**

Le protocole PPTP (*Point To Point Tunneling Protocol*) est d'origine Microsoft, il est détaillé par la RFC 2637. Une trame PPP (*Point To Point Protocol*) est transportée sur UDP à l'aide du protocole GRE (*Generic Routing Encapsulation*) développé par CISCO. Le protocole PPP utilise par exemple EAP (*Extensible Authentication Protocol*) pour l'authentification la génération de clés de chiffrement. La double procédure EAP PEAP/MSCHAPv2 soit l'établissement d'un premier tunnel TLS, puis un échange de messages selon MSCHAPv2 protégeant un mot de passe est largement utilisée dans ce contexte.

### Limitation des protections offertes par les pare-feu.



Un pare-feu<sup>4</sup> filtre les paquets IP échangés avec un intranet en analysant l'entête d'un paquet IP, celui du protocole transporté, et parfois certains paramètres de l'application (mode proxy ou à états). Il y a un compromis entre la sévérité du filtrage (paquets IP fragmentés, paquets ICMP ou UDP) et le degré de fonctionnalité/convivialité que l'on veut obtenir. Cependant un pare-feu ne protège pas des attaques internes, par exemple un cheval de Troie peut utiliser une session TCP pour échanger de l'information par émission de données à l'extérieur des fenêtres de réception du destinataire, ou par réémissions de segments déjà acquittés.



De manière analogue aux forteresses médiévales, on peut créer une zone démilitarisée (DMZ) entre deux pare-feu, la DMZ accueille les serveurs

<sup>4</sup> Chuck Semeria, "Internet Firewalls and Security", 1996

d'information de l'entreprise et un Bastion, un serveur spécialement sécurisé pour la gestion d'information critiques accessibles depuis l'extérieur. Les réseaux externe et externe accèdent au Bastion et au serveur d'information. Le Bastion peut par exemple gérer le NAT du réseau interne.

### De la faiblesse des systèmes informatiques.

Trois facteurs<sup>5</sup> (ou *Trinité*) amplifient les failles des systèmes informatiques modernes,



- **La complexité.** Les logiciels sont complexes, les développeurs ne maîtrisent pas les *bugs* et les comportements non désirés.

- **L'extensibilité.** La configuration d'une plateforme informatique se modifie tout au long de sa durée de vie.

- **La connectivité.** Les vulnérabilités des logiciels sont exploitables à distance.

### Faites moi confiance ? 20 ans de bugs et heuristiques

Voici une liste non exhaustive de failles de sécurité

- 1994, Peter Shor invente un algorithme de factorisation d'un nombre  $N$  par un ordinateur quantique, en un temps  $O((\log N)^3)$ . En d'autres termes RSA est cassable par une technologie quantique. IBM a factorisé le nombre 15 ( $3 \times 5$ ) en utilisant un ordinateur quantique de 7 qbits.

- 1996, Lov K. Grover<sup>6</sup> publie un algorithme quantique permettant une recherche exhaustive dans un espace à  $N$  éléments en  $O(\sqrt{N})$ . En particulier une clé de 128 bits peut être cassée selon une complexité de  $2^{64}$ . En 2013 Google associé à la NASA a fait l'acquisition d'un ordinateur quantique (D-WAVE).

- 1998, l'algorithme COMP128-1, qui assure l'authentification des abonnés du GSM (algorithme A3/A8) est craqué en  $2^{19}$  (0,5 million) essais par l'université de Berkeley<sup>7</sup>.

<sup>5</sup> S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in Embedded Systems: Design Challenges", 2004.

<sup>6</sup> Lov K. Grover "A fast quantum mechanical algorithm for database search", 1996

<sup>7</sup> Marc Briceno, Ian Goldberg, and David Wagner

- 1999, Serge Humpich casse la clé RSA privée de 320 bits des cartes bancaires, à l'aide d'un logiciel téléchargé sur le WEB.

- 2001, Fluhrer & All<sup>8</sup> cassent le WEP en environ  $2^{22}$  (4 millions) d'essais.

- 2002, trois scientifiques indiens<sup>9</sup> Manindra Agrawal, Neeraj Kayal et Nitin Saxena publient le test de primalité AKS, qui a reçu le prix Gödel en 2006. La complexité de l'algorithme est en  $O((\log n)^{12})$ , soit par exemple pour un nombre de 1024 bits  $O(2^{120})$ . D'où la difficulté dans un protocole d'avoir une preuve de la primalité d'un entier. Une variante de AKS de H. W. Lenstra, Jr. et Carl Pomerance (2005) réduit la complexité à  $O((\log n)^6)$  soit  $O(2^{60})$  pour un nombre de 1024 bits.

- Mai 2003, Muhammad Faisal Rauf Danka, (étudiant pakistanais) accède à tous les comptes Passport.Net, en insérant la chaîne emailpwdreset dans une URL d'accès,

<https://register.passport.net/emailpwdreset.srf?lc=1033&m=victim@hotmail.com&id=&cb=&prefem=attacker@attacker.com&rst=1>.

- Juillet 2003, Jérôme Cretaux met en évidence l'absence de confidentialité quant aux informations sensibles, stockées dans les cartes Sésame Vitale

- 2004, effondrement de la probabilité de collision de l'algorithme MD5 développé dans les années 90, et largement utilisé dans l'internet, dont la valeur théorique était de  $1/2^{80}$  (soit un temps d'attaque "infini")

- Wang et al (2004)  $1/2^{39}$  (temps d'attaque 1 heure)

- Marc Stevens (2006)  $1/2^{32}$  (temps d'attaque 5 mn)

- 2005, premier clonage de certificats X509, utilisant des signatures MD5 (Arjen Lenstra).

- 2008, Karsten Nohl, doctorant à *Virginia University* casse la carte Mifare Crypto1, vendue à plus de 500 millions d'exemplaires et utilisée par exemple comme carte de transport (Oyster card Londres, OV-chipkaart Pays-Bas, Charlie card Boston). La sécurité de ce composant RFID repose sur une clé secrète de 48 bits et un registre à décalage de type *Fibonacci LFSR*.

- en 2008, Jeroen van Beek, modifie avec succès le RFID des passeports électroniques Hollandais. L'attaque repose sur deux concepts, réalisation

---

<sup>8</sup> S. Fluhrer, I. Mantin, and A. Shamir. "Weaknesses in the key scheduling algorithm of RC4". In *Eighth Annual Workshop on Selected Areas in Cryptography*, Toronto, Canada, Aug. 2001.

<sup>9</sup> Manindra Agrawal, Neeraj Kayal and Nitin Saxena "PRIMES is in P", 2002

d'un clone logiciel et exploitation d'incohérences des spécifications d'interopérabilité (une signature incorrecte est une erreur non critique, une valeur de hash erronée est un avertissement).

- en 2010, un groupe de chercheurs anglais<sup>10</sup>, mettent en évidence l'absence de vérification du code PIN dans certaines cartes bancaires EMV, lors de transactions de paiement.

- en 2011 le système RSA SecurID a été victime d'une attaque dite APT (pour *Advanced Persistent Threat*), c'est-à-dire un malware variante du logiciel *Poison Ivy*. Un jeton SecurID possède un numéro de série et stocke une clé secrète de 128 bits, usuellement nommée *seed*. Il génère un *code* basé sur une fonction de hash, la date, et le *seed*,  $code=h(date || seed)$ . L'utilisateur connaît un PIN associé au jeton. Le *passcode* est la concaténation du code affiché par le jeton et du PIN utilisateur. Une base de données stocke les tuples *seed*, numéro de série ainsi que le login utilisateur. Il est possible (?) que le *seed* soit une fonction du numéro de série. L'attaque a (?) permis d'obtenir tout ou partie de la base de données. Un avertissement de la société indique qu'il serait possible de récupérer le *seed* à partir d'un seul passcode.

- en 2013 Karsten Nohl a présenté lors de la conférence, Black Hat 2013<sup>11</sup>, une attaque permettant de «rooter» une carte SIM à partir d'un seul SMS, c'est-à-dire d'obtenir une clé DES de 56 bits dont la connaissance permet de charger, d'activer ou de détruire des applications embarquées (dans la SIM). Cette attaque s'applique à environ 1/8 du parc des cartes SIM. Elle repose sur un «bug» logiciel. La réception d'un message contenant un CMAC DES d'authentification, dont le contenu est aléatoire, implique la génération d'un message contenant un CMAC d'authentification calculé avec une clé DES 56 bits légitime. Le craquage force brute de cette clé utilise la technique dite des "*rainbow tables*".

---

<sup>10</sup> Murdoch, Steven J.; Drimer, Saar; Anderson, Ross; Bond, Mike; "Chip and PIN is Broken", Security and Privacy (SP), 2010 IEEE Symposium on Digital Object Identifier: 10.1109/SP.2010.33

<sup>11</sup> Karsten Nohl, "Rooting SIM cards", Black Hat 2013, <https://media.blackhat.com/us-13/us-13-Nohl-Rooting-SIM-cards-Slides.pdf>

- en 2014 un groupe<sup>12</sup> de chercheurs français publie un algorithme quasi polynomial réalisant le calcul d'un logarithme discret dans les corps finis, de complexité

$$2^{O((\log \log Q)^2)}$$

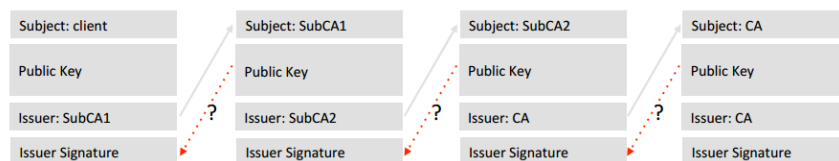
Pour des corps fini tels que

$$\mathbb{F}_Q = \mathbb{F}_{q^{2k}} \text{ avec } k \# q$$

ou pour des corps de faibles caractéristiques (typiquement 2)

$$\mathbb{F}_{2^n} \text{ soit une complexité de } 2^{O((\log n)^2)}$$

- En 2014 Jeff Forristal directeur technique de la société Bluebox a publié lors de la conférence<sup>13</sup> Blackhat 2014 une attaque Android nommé "Android Fake ID". Le système d'exploitation utilise des chaines de certificats dont la racine est un certificat auto-signé pour la vérification des signatures. Cependant la signature des certificats à l'intérieur de la chaîne n'est pas vérifiée.



- En 2014 Apple a publié une faille de sécurité dans le logiciel OPENSSL utilisé dans l'iPhone, l'iPad et le Mac OS X; la signature des certificats n'est pas vérifiée. Le code source responsable du problème est listé ci dessous.

```
static OSStatus
```

```
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa,
SSLBuffer signedParams,
```

<sup>12</sup> Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé, "A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic".

<sup>13</sup> <https://www.blackhat.com/docs/us-14/materials/us-14-Forristal-Android-FakeID-Vulnerability-Walkthrough.pdf>

```

uint8_t *signature, UInt16 signatureLen)
{
  OSStatus err;
  ...
  if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
  if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
  goto fail;
  if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
  ...
  fail:
  SSLFreeBuffer(&signedHashes);
  SSLFreeBuffer(&hashCtx);
  return err;
}

```

### Quelques paradigmes de défense.



- **Le placebo.** Une défense utopique mais parfois efficace, est utilisée pour parer une attaque inconnue. Par exemple la saignée, l'emploi hasardeux d'antibiotiques ou de vaccin.



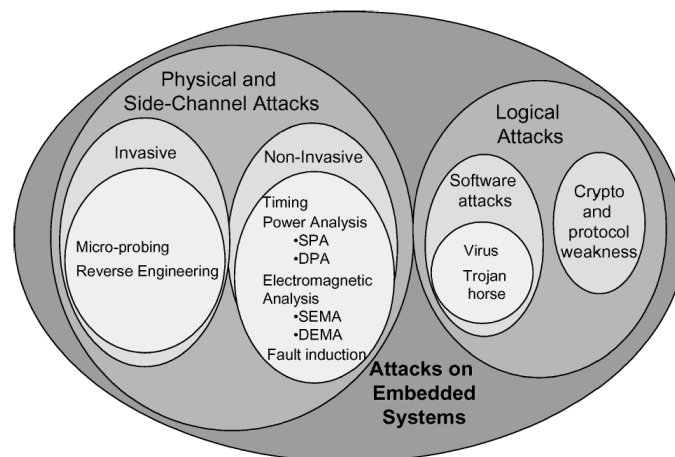
- **Le vaccin.** Une réponse connue et efficace à une attaque identifiée.



- **La défense immunitaire.** Une réponse efficace, est spontanément générée, pour lutter contre une attaque inconnue. La légende d'*Hans Brinker* est une illustration de ce principe.



### Exemple de classification des attaques pour les systèmes embarqués.



S. Ravi et al, 2004      Exemples of attack threats faced by embedded systems.

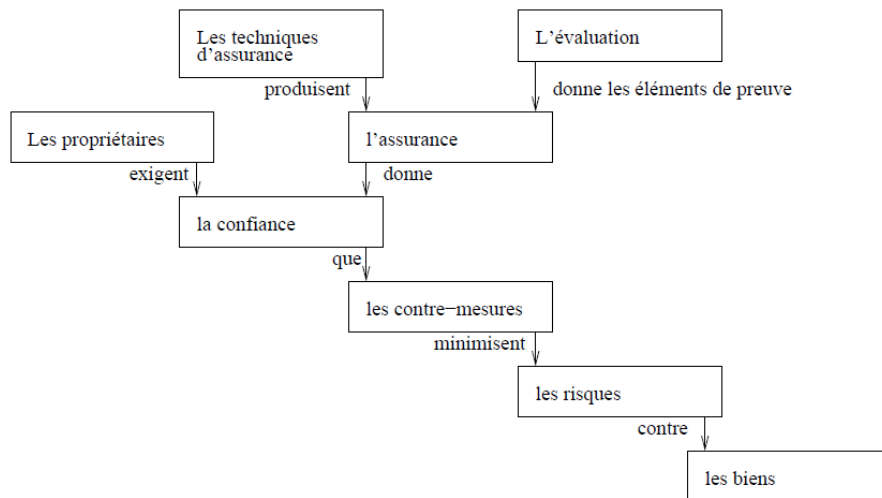
### Au sujet des normes Critères Communs (CC)

Les normes ISO 15408 dénommées *Critères Communs* visent à qualifier la sécurité de produits tels que microcontrôleurs sécurisés, cartes à puce ou firewalls. Elles sont organisées en trois parties, introduction (partie 1), exigences fonctionnelles de sécurité (partie 2) et exigences d'assurances de sécurité (partie 3).

Une évaluation CC est une procédure qui décerne un EAL (*Evaluation Assurance Level*) selon un document de référence, la cible de sécurité (*Security Target*).

La cible de sécurité liste les *exigences fonctionnelles* de sécurité (CC partie 2, par exemple identification et authentification - classe FIA) et les *exigences d'assurance de qualité* (CC partie 3, par exemple les tests - classe ATE) du produit, c'est-à-dire le TOE (*Target Of Evaluation*) dans la terminologie CC.

Les exigences fonctionnelles et d'assurances sont classées selon une hiérarchie à trois niveaux: classe-famille-composants. De manière optionnelle le TOE peut être décrit dans un *Protection Profile* (ou PP) qui est commun à un ensemble de produits dont les fonctionnalités et les exigences de sécurité sont similaires.



Les niveaux EAL se répartissent en sept catégories

- EAL 7 conception formelle vérifiée et produit testé
- EAL 6 conception semi-formelle vérifiée et produit testé
- EAL 5 produit conçu de façon semi-formelle et testé
- EAL 4 produit conçu, testé, revu de façon méthodique
- EAL 3 produit testé et vérifié de façon méthodique
- EAL 2 produit testé structurellement
- EAL 1 produit testé fonctionnellement

Le niveau augmenté (noté +) indique que lors de l'évaluation des informations complémentaires, telles que les codes sources, sont connues.

Les niveaux de produit bancaire sont par exemple EAL4+ ou EAL5, des composants logiciels complexes EAL1.

## L'intrusion.



L'intrusion consiste à obtenir des autorisations (privilèges) illicites sur un système informatique afin d'accéder à des ressources (fichiers) ou de contrôler certaines ressources (accès réseaux...). Elle peut être menée par un utilisateur licite d'un système ou à l'aide de sessions réseaux. L'intrusion non autorisée de logiciel peut prendre la forme d'un virus, d'un ver, ou d'un cheval de Troie.

Anderson a proposé dans les années 80 une classification à trois niveaux

- **La mascarade.** Exploitation illégitime d'un compte utilisateur (cassage de mot de passes ...)
- **L'usurpation.** Un utilisateur légitime qui étend ses privilèges.
- **L'usage clandestin.** Le contournement des barrières de sécurité, la prise de contrôle du système.

La détection des intrusions est basée sur des collectes d'informations (audit, sondes réseaux...), sur la détection d'anomalies statistiques (moyenne, écart type, filtres bayésien...) ou la corrélation d'évènements (signature des attaques réseaux à l'aide de formules booléennes...).

## Programmes malveillants, virus, vers

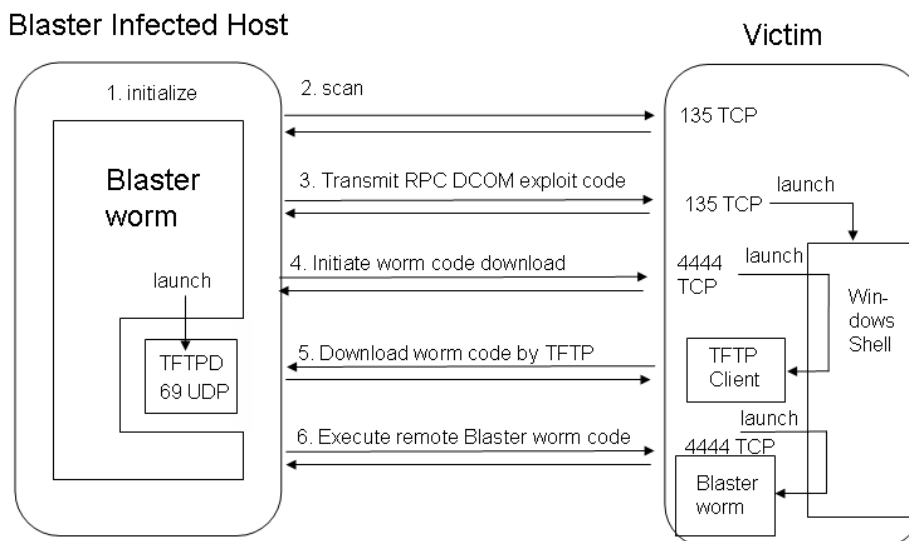
Les programmes malveillants sont de multiples natures, portes dérobées (backdoors, fonctionnalisées cachées), bombes logiques, chevaux de Troie, Virus, Vers.



Un virus est un programme qui se duplique, on distingue classiquement quatre phases : la phase dormante, la propagation, le déclenchement, l'exécution. Il se caractérise par un code spécifique et un comportement (actions) typique sur un système informatique.

Un vers est un programme offensif qui se propage à travers le réseau.

### Exemple le ver blaster



Principe du ver «Blaster» (2003)

### Au sujet des botnets

Le terme botnet<sup>14</sup> est la contraction des mots "roBOT" et "NETwork", soit littéralement réseau de robots. Un robot est un programme effectuant des actions automatiques sur un serveur. Les premiers robots ont vu le jour dans le monde des opérateurs IRC. La dérive fut d'utiliser ces outils d'administration afin de prendre l'avantage lors de compétition pour le contrôle d'un canal IRC. Le plus connu de ces outils est *Eggdrop*, apparu en 1993. Le premier botnet *W32/Pretty.worm* ciblait des systèmes Windows dans les années 2000.

Un botnet contamine un parc de systèmes informatiques, qualifiés de *zombies*. Le nombre de machines impliqués peut être de l'ordre de plusieurs millions, par exemple pour le botnet *Zeus*. Il se propage selon différentes techniques telles failles logicielles de systèmes d'exploitation, scripts ou programmes malveillants. Il utilise des canaux de commandes variés par exemple réseaux P2P ou échanges HTTP. Les botnets sont utilisés pour générer des SPAMs ou des attaques de type dénis de service (DDOS).

<sup>14</sup> F. Ducrot, M. Danho, X. Marronnier, SÉCURITÉ INFORMATIQUE, numéro 61, CNRS, octobre 2007

## Au sujet des rootkits

Un *rootkit* est un logiciel malveillant furtif, c'est-à-dire qu'il masque son activité dans un système d'exploitation (logs, liste des processus actifs, communications réseaux ...). Il peut s'installer au sein d'un autre logiciel, une bibliothèque ou dans le noyau d'un système d'exploitation. Les premiers rootkits sont apparus en 1994 sur Linux et SunOS4; en 1998, sous Windows, (Back Orifice) et en 2004 sous Mac OS (X, WeaponX5).

## Quelques techniques malveillantes classiques.

### *Le buffer overflow*

Le *buffer overflow* consiste à modifier malicieusement la mémoire d'un programme, typiquement lors de l'écriture d'une information localisée dans un paramètre d'appel du programme, ou lors de l'écriture de données reçues via le réseau (c'est-à-dire à l'aide des bibliothèques de sockets). Le langage C utilise des chaînes (`char[]`) de caractères dont la marque de fin est un octet nul. Les fonctions qui manipulent ces chaînes (notées avec un préfixe `s`, par exemple `strcpy,...`), peuvent modifier une plage mémoire supérieure à la taille allouée par le concepteur du programme. Par exemple

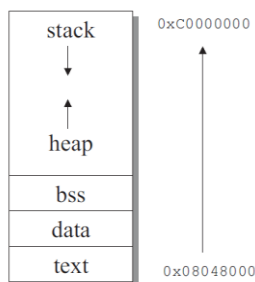
```
char name[12] ;
char *information; // la marque de fin est un octet nul
strcpy(name, information) ;
```

Si la taille de *information* dépasse 12 octets, la mémoire allouée pour *name* est insuffisante, en conséquence une partie du contenu du pointeur *information* modifie la mémoire du programme de manière imprévue.

Les effets espérés par les *buffer overflow* dans la pile d'exécution sont répartis en trois classes principales

- La *modification d'une variable mémoire*, proche de la zone mémoire occupée par le buffer.
- la *modification de l'adresse de retour du programme* localisée dans la pile, le but étant l'exécution d'un code malveillant contenu dans un paramètre d'appel ou dans une information reçue via le réseau.
- la *modification d'un pointeur de fonction* afin d'exécuter un code préalablement injecté.

Les effets des *buffer overflow* dans la zone mémoire dite *heap* (ou tas) sont différents, car cette dernière est utilisée par les fonctions de type *malloc* réalisant les allocations dynamiques de mémoire.



Par exemple le format ELF (*Executable Linking Format*), utilisé par de nombreux systèmes UNIX, définit cinq zones de mémoires virtuelles pour un programme, *text* pour le code, *data* pour les données initialisées, *bss* pour les données non initialisées, la pile (*stack*) et le tas (*heap*) qui est la zone mémoire restante intercalée entre pile et données.

Le *shellcode* désigne un programme qui exécute un ensemble de commandes du *shell*. Il est possible d'exécuter un *shell* dans un programme, à l'aide d'un appel système spécifique; un code malicieux exploite cette fonctionnalité. Une attaque consiste dans ce cas à injecter un *shellcode* puis à l'exécuter via des techniques de *buffers overflow*.

En 1988, un étudiant du MIT, Robert Tappan Morris, réalisa la première mise en œuvre offensive sur internet des techniques *buffer overflow*, nommée *Morris worm*.

### Le fuzzing

On peut également citer les techniques **fuzzing** d'attaques de protocoles réseaux dont le principe est d'injecter des données aléatoires dans les logiciels de traitement des piles (*stack*) de communication.

### L'injection SQL

L'**injection SQL** consiste à importer des données qui sont des instructions du langage de requêtes SQL. Par exemple un script PHP intègre directement dans une requête SQL, un login et un mot de passe renseignés dans un formulaire (HTML). Puisque ces paramètres sont des éléments du langage, ils peuvent modifier malicieusement la requête; il devient en conséquence possible de valider un mot de passe erroné.

### Le Cross Site Scripting

Le **Cross Site Scripting** (XSS) consiste à injecter un script (javascript, applet, object) dans un formulaire HTML, dans un contexte où l'application WEB affiche des contenus interactifs (blogs, etc...) à ses utilisateurs. Le

script posté par l'attaquant sera par la suite téléchargé puis exécuté par les navigateurs des clients (victimes) de l'application WEB.

### Un exemple de stratégie de défense.

Voici un bref résumé de la stratégie de sécurité d'un grand éditeur de logiciels (Microsoft).

- Conception de systèmes d'exploitation résistant aux attaques (*resilency*), en particulier en introduisant la notion de comportements (*behavior*) basés sur des politiques d'usage de services.
- Usage généralisé d'IPSEC et des VPNs pour les liens interentreprises.
- Introduction des critères communs (*quality*) pour la conception de programme (*Safe Programming*).!
- Généralisation de l'usage des pare-feu applicatifs.
- Lutte contre le SPAM (?)
- Concept du SD3+C (*Secure by Design, Default, Deployment, Communications*)
- Authentification à deux facteurs des utilisateurs, à l'aide de cartes à puce ou de jetons.
- Participation des utilisateurs à la politique de sécurité grâce à l'éducation ou à la répression.

### Intégrité du code, obfuscation, TPM

Certaines attaques intrusives sont liées à la possibilité de modifier le code d'un programme soit avant son exécution (modification du fichier exécutable) soit au cours de son exécution (point d'arrêts en mode *debug*, modification de certaines zones mémoire, pose de bretelles....).

L'analyse d'un code lors de son exécution révèle la clé associée à un algorithme cryptographique. Cependant des techniques d'embrouillage de code (*obfuscation*) peuvent rendre ces attaques plus difficiles.

Le groupe *Trusted Computing Group* (TCG) a défini une architecture sécurisée pour les ordinateurs personnels basée sur un module hardware sécurisé le TPM (*Trusted Platform Module*). Dans ce modèle l'intégrité du système (bibliothèques essentielles du système d'exploitation,...) est mesurée (*integrity measurement*) par une empreinte (SHA-1, 160 bits) stockée

dans une puce de silicium résistante aux attaques (*tamper resistant*). L'accès à ce dispositif est contrôlé par des secrets partagés symétriques. Le TPM s'appuie sur un arbre de clés RSA, dont l'accès à chaque nœud est protégé par une clé symétrique. Remarquons à ce propos qu'une clé RSA de modulo  $N$  peut chiffrer les paramètres d'une autre clé RSA de modulo  $n=pq$  et d'exposants public et privé  $d,e$  tels que  $p < N$ ,  $q < N$ ,  $d < N$ ,  $e < N$ .

L'intégration du TPM dans le système d'exploitation *Windows* s'applique aux trois points suivants,

- Le *Secure Boot*. Le premier programme amorce est stocké dans le TPM. Le *boot* est une succession de programme  $P_i$  tels que  $P_0$  est contenu dans le TPM, chaque  $P_i$  est associé à une empreinte  $H_i$  enregistrée dans le TPM, le programme  $P_{i-1}$  charge  $P_i$  et vérifie son empreinte  $H_i$ .
- Le chiffrement du contenu du disque dur (*bitlocker<sup>TM</sup>*) à l'aide d'une clé maître (VEK, Volume Encryption Key) stockée dans le TPM.
- Le contrôle de l'intégrité des PCs au moment de leur connexion réseau (*NAP, Network Access Protection*).



## Les Canaux Cachés.



Bien que les algorithmes cryptographiques tels que RSA ou DES soient considérés comme sûres d'un point de vue mathématiques, il est possible d'extraire les clés à partir des temps d'exécution ou des ondes rayonnées par le processeur qui réalise ces fonctions. Nous donnons ici deux exemples de telles attaques (SPA – Single Power Analysis & DPA – Differential Power Analysis, *Paul Kocher*).

### *Single Power Analysis*

Un calcul RSA consiste à élever un nombre  $M$  (le message en clair) à la puissance  $e$  en modulo  $m$  (le couple  $e$  et  $m$  constituant la clé privé), soit encore

$$C(\text{forme chiffrée}) = M^e \text{ modulo } m$$

En exprimant  $e$  sous forme binaire soit,

$e = e_0.2^0 + e_1.2^1 + e_2.2^2 + e_3.2^3 + e_4.2^4 + \dots + e_i.2^i + \dots + e_{p-1}.2^{p-1}$ , ou  $e_i$  a pour valeur 0 ou 1.

La forme chiffrée s'exprime sous forme d'un produit de  $p$  termes  $m_i$ ,

$C = m_0. m_1 m_2 \dots m_i \dots m_{p-1}$  modulo  $m$ , avec

- $m_i = 1$ , si  $e_i = 0$ .
- $m_i = M^{2^i}$  modulo  $m$ , si  $e_i = 1$

En constate que, dans cette implémentation de l'algorithme RSA (dite *exponentiation*), chaque bit ( $e_i$ ) de la clé implique un temps calcul différent selon que sa valeur soit 0 (multiplication triviale par 1) ou 1 (multiplication par  $M^{2^i}$ ). En fonction des différences de temps calculs observées on déduit la valeur de  $e_i$  (0 ou 1).

### *Differential Power Analysis*

Nous avons vu précédemment que dans le cas de RSA, le calcul peut être réalisé en  $p$  étages, la taille de l'exposant privé étant de  $p$  bits.

Supposons un algorithme cryptographique dont le calcul se décompose en  $p$  étages. Chaque étage  $k$  (étant compris entre 0 et  $p-1$ ) prend comme argument d'entrée une valeur  $M_{k,i}$  et calcule la valeur  $M_{k+1,i}$ .

$M_{0,i}$  est le message en clair numéro  $i$ , et  $M_{p,i}$  sa forme chiffrée.

Un microprocesseur qui exécute un calcul induit des effets physiques variés, par exemple l'énergie consommée ou l'émission d'ondes radio. Nous désignons par  $S_{k,i}(t)$  un effet physique produit par l'étage de calcul  $k$  relativement au message  $i$ .

De surcroît nous imposons que chaque étage  $k$  utilise une clé  $K_{k,j}$  dont la taille est de  $n_k$  bits ( $j$  est une valeur comprise entre 0 et  $2^{n_k} - 1$ ).

Par exemple dans le cas de l'algorithme DES chaque étage utilise une (sous) clé de 6 bits ( $n_k = 6$ ). L'algorithme travaille avec une clé globale de 56 bits, mais cette dernière est appliquée sur différents blocs de calculs associés à des clés de 6 bits.

Nous supposons l'existence pour chaque étage  $k$  d'une fonction  $g_{k,j}(M_{k,i})$  que nous nommons estimateur, telle que pour chaque clé  $K_{k,j}$ , la moyenne relativement aux valeurs d'entrée ( $M_{k,i}$ ) est nulle, c'est à dire que pour toute clé  $j$  ( $j$  compris entre 0 et  $2^{n_k}-1$ ).

$$1/N \times \sum_{0 \leq i < N-1} g_{k,j}(M_{k,i}) \sim 0 \text{ pour } N \text{ très grand}$$

Pour fixer les idées nous enregistrons par exemple la puissance consommée  $S_{k,i}(t)$  par le microprocesseur au cours du calcul de l'étage  $k$  pour une valeur d'entrée  $M_{k,i}$ .

Nous calculons pour chaque relevé l'ensemble des  $2^{n_k}$  produits  $P_{k,j,i}(t)$

$$P_{k,j,i}(t) = S_{k,i}(t) g_{k,j}(M_{k,i})$$

On obtient  $2^{n_k}$  courbes variant dans le temps, pour chaque étage  $k$  et pour chaque message  $i$ .

Si l'on admet que pour les mauvaises clés les fonctions  $S_{k,i}(t)$  et  $g_{k,j}(M_{k,i})$  ne sont pas corrélées au sens statistique du terme, alors la moyenne de leur produit est égal au produit des moyennes et donc est proche de 0.

$$1/N \sum_{0 \leq i < N-1} P_{k,j,i}(t) = 0(t) \text{ pour } N \text{ très grand et } j \text{ mauvaise clé}$$

En revanche si l'on admet que pour la bonne clé les fonctions  $S_{k,i}(t)$  et  $g_{k,j}(M_{k,i})$  sont corrélées au sens statistique du terme, alors la moyenne de leur produit n'est pas égale au produit des moyennes, et donc, n'est pas nulle.

$$1/N \sum_{0 \leq i < N-1} P_{k,j,i} \neq 0(t), \text{ pour } N \text{ très grand et } j \text{ bonne clé}$$

## Récentes attaques sur les fonctions de hash MD5 et SHA-1

Une fonction d’empreinte  $H$  produit, à partir d’un message  $M$  une valeur pseudo aléatoire de  $p$  bits (soit  $2^p$  empreintes). Les attaques sont classées en trois catégories :

- **Collision**: Trouver un couple  $(M, M')$ , tel que  $H(M) = H(M')$
- **1<sup>st</sup> pré-image**, étant donné  $X$ , trouver  $M$  tel que  $H(M) = X$
- **2<sup>nd</sup> pré-image**, étant donné  $M$ , trouver  $M'$  tel que  $H(M') = H(M)$

Dans le cas d’un algorithme parfait, la probabilité d’une *collision* est de  $1/2^{p/2}$  (en raison du paradoxe des anniversaires) et pour les *pré-images*  $1/2^p$ .

En 2004 des collisions MD5 ont été produites avec un coût de  $2^{39}$ , et des collisions SHA1 avec un coût théorique de  $2^{69}$ . “A paper by Xiaoyun Wang and Dengguo Feng and Xuejia Lai and Hongbo Yu has been posted on Aug 17, 2004 about Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, showing collisions for the MD5 hash with the right input vectors”.

En 2005 *Lenstra et al* ont forgé à partir d’un premier certificat X509 (signature MD5) un second certificat qui conserve la signature du premier mais comporte une autre clé RSA publique.



En 2008, Stevens et al («*Chosen-prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities*») ont introduit une méthode de collision MD5 avec préfixe choisi, qui réalise une collision en environ  $2^{50}$  essais. Cet algorithme appliqué sur une plateforme comportant 200 *plays station*, permet de cloner un certificat MD5 en **un à deux jours**.

En 2009, lors de la conférence Eurocrypt’2009, Cameron McDonald, Philip Hawkes et Josef Pieprzyk, ont annoncé un algorithme de collision de SHA1 d’une complexité de  $2^{52}$ .

## Quelques TOP10 d'attaques

### *Le TOP9 des menaces visant le cloud computing en 2013*

La virtualisation est une technologie inventée par IBM dans les années 70, puis déployée par exemple par Microsoft pour Windows95. Cette

technologie est basée sur des hyperviseurs capables de gérer plusieurs machines virtuelles. On distingue les hyperviseurs hébergés ("*hosted*") exécutés par un système d'exploitation, ou natifs dans le cas contraire.

Le rapport<sup>15</sup> "*The Notorious Nine Cloud Computing Top Threats in 2013*" publié en 2013 par le Cloud Security Alliance établit une liste des 9 attaques les plus significatives du Cloud.

1) La fuite de données. Une équipe de chercheurs américains a démontré une attaque DPA (*Differential Power Analysis*) entre des machines virtuelles permettant de récupérer des clés cryptographiques privées ("*Cross-VM Side Channels and Their Use to Extract Private Keys*", Yinqian Zhang, Ari Juels, Thomas Ristenpart, Michael K. Reiter, 2012). Le rapport souligne également la possibilité d'extraction d'informations à partir d'un trou de sécurité dans une application cliente de base de données multi parties (*multi-tenant*).

2) La perte de données. La perte de données peut résulter de problèmes matériels du fournisseur de Cloud, de malveillance (vol de mot de passe, intrusion...) ou de sinistres divers (incendie, inondation, tempêtes,...).

3) Détournement de comptes. En avril 2010, une faille de type Cross-Site Scripting (XSS) a permis à des hackers de récupérer des identifiants de compte Amazon. En 2009, le *botnet* Zeus s'est propagé sur de nombreux systèmes Amazon.

4) Interfaces et APIs non sécurisés. L'accès aux ressources du cloud implique un contrôle d'accès et des politiques de sécurité basés sur des identifiants. La collecte de ces identifiants via des interfaces ou APIs présentant des failles de sécurité permet le détournement de comptes utilisateurs ou l'abus de privilèges.

5) Le dénis de service. Les attaques DDOS (*distributed denial-of-service*) typiquement menées à l'aide de botnets peuvent être internes ou externes au cloud. Une attaque DDoS impacte l'ensemble de clients dont les services sont hébergés par un cloud.

6) Délit d'initié. C'est typiquement un employé ou un ancien employé du fournisseur de Cloud qui accède à des fins malveillantes, aux ressources hébergées.

---

15

[https://downloads.cloudsecurityalliance.org/initiatives/top\\_threats/The\\_Notorious\\_Nine\\_Cloud\\_Computing\\_Top\\_Threats\\_in\\_2013.pdf](https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf)

7) Abus de services. Les services du cloud peuvent être utilisés à des fins malhonnêtes, par exemple pour casser des clés cryptographiques ou héberger des sites illégaux (P2P,...)

8) Manque de compétences. Certaines entreprises n'appréhendent pas suffisamment les compétences requises, notamment en termes de sécurité, pour réaliser une migration de leurs ressources informatiques dans le cloud.

9) Vulnérabilités technologiques. Les hyperviseurs peuvent s'appuyer sur des processeurs présentant des failles de sécurité telles que, escalade de privilèges, ou défaut d'isolation mémoire (Memory Management Unit - MMU).

### **OWASP TOP10 2013**

Open Web Application Security Project (OWASP, [www.owasp.org](http://www.owasp.org)) est une communauté publique permettant à des organismes de développer, acheter et maintenir des applications fiables. Il publie<sup>16</sup> chaque année un rapport sur dix les risques de sécurité WEB les plus critiques(())

A1- Faille d'injection. Par exemple 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.

A2 - Violation de Gestion d'Authentification et de Session. Les fonctions applicatives relatives à l'authentification et la gestion de session ne sont souvent pas mises en œuvre correctement, permettant aux attaquants de compromettre les mots de passe, clés, jetons de session, ou d'exploiter d'autres failles d'implémentation pour s'approprier les identités d'autres utilisateurs.

A3 - Cross-Site Scripting (XSS). Les failles XSS se produisent chaque fois qu'une application accepte des données non fiables et les envoie à un browser 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, de modifier des sites web, ou rediriger l'utilisateur vers des sites malveillants.

A4 - 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

---

<sup>16</sup> [http://www.owasp.org/index.php/Top\\_10](http://www.owasp.org/index.php/Top_10)

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.

A5 - Mauvaise configuration Sécurité. • Une bonne sécurité nécessite de disposer d'une configuration sécurisée définie et déployée pour l'application, contextes, serveur d'application, serveur web, serveur de base de données et la plate-forme. Tous ces paramètres doivent être définis, mis en œuvre et maintenus, car beaucoup ne sont pas livrés sécurisés par défaut. Cela implique de tenir tous les logiciels à jour.

A6 - Exposition de données sensibles. Beaucoup d'applications web ne protègent pas correctement les données sensibles telles que les cartes de crédit, identifiants d'impôt et informations d'authentification. Les pirates peuvent voler ou modifier ces données faiblement protégées pour effectuer un vol d'identité, de la fraude à la carte de crédit ou autres crimes. Les données sensibles méritent une protection supplémentaire tel un chiffrement statique ou en transit, ainsi que des précautions particulières lors de l'échange avec le navigateur.

A7 - Manque de contrôle d'accès au niveau fonctionnel. Pratiquement toutes les applications web vérifient les droits d'accès au niveau fonctionnel avant de rendre cette fonctionnalité visible dans l'interface utilisateur. Cependant, les applications doivent effectuer les mêmes vérifications de contrôle d'accès sur le serveur lors de l'accès à chaque fonction. Si les demandes ne sont pas vérifiées, les attaquants seront en mesure de forger des demandes afin d'accéder à une fonctionnalité non autorisée.

A8 - Falsification de requête inter-site (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 incluse, 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.

A9 - Utilisation de composants avec des vulnérabilités connues. Les composants vulnérables, tels que bibliothèques, contextes et autres modules logiciels fonctionnent presque toujours avec des privilèges maximum. Ainsi, si exploités, ils peuvent causer des pertes de données sérieuses ou une prise de contrôle du serveur. Les applications utilisant ces composants vulnérables peuvent compromettre leurs défenses et permettre une série d'attaques et d'impacts potentiels.

A10 – Redirections et renvois non validés. Les applications web réorientent et redirigent fréquemment les utilisateurs vers d'autres pages et sites internet, et utilisent des données non fiables pour déterminer les pages de destination. Sans validation appropriée, les attaquants peuvent réorienter les victimes vers des sites de phishing ou de malware, ou utiliser les renvois pour accéder à des pages non autorisées.

### Principes de sécurité du système Android

Android est un système d'exploitation créé par la société Android Inc. rachetée en 2005 par GOOGLE. Il est aujourd'hui largement déployé dans les mobiles.

Le système Android s'appuie sur un noyau UNIX. L'image binaire du système nommée ROM, stocke le noyau, les bibliothèques natives, l'environnement de la Dalvik Virtual Machine (DVM), le framework Java (JNI), et les applications.

Une application Android comporte au plus quatre composants : l'activité, le service, le fournisseur de contenu et le receveur de *broadcast*. Les composants peuvent être déclarés publics ou privés. Les composants activités, services, et broadcast receivers, sont activés par un message asynchrone dénommé *Intent*.

La sécurité android repose sur trois piliers :

- Les SANDBOXs Unix, associés aux processus
- Les permissions.
- La signature des applications

Un SANDBOX est un environnement logiciel qui contrôle les accès d'une application aux ressources d'un système informatique géré par un système d'exploitation.

Pour le noyau Linux associé à Android, une application possède un user-id, un id de groupe et un id de groupe secondaire. Chaque application possède des droits d'accès en lecture, écriture et exécution. C'est une politique de contrôle d'accès discrétionnaire (DAC).

Le système Unix possède un compte *root* dont le user-id est 0, et qui possède tous les droits. Chaque Application possède sa propre instance de Dalvik Virtual Machine (DVM). Deux applications signées par une même clé privée peuvent partager un Sandbox identique.

Le fichier *Manifest.xml* définit la structure d'une application, c'est-à-dire la liste de ses composants. Il contient également les demandes de permission d'accès aux ressources du mobile et décrit les *Intents* traités par l'application. C'est le gestionnaire de la politique de sécurité de l'application.

Chaque application (.apk) doit être signée. Sous Eclipse on utilise les outils *Keytool* pour la génération de clés RSA dans un magasin de certificats (keystore), et *Jarsigner* pour la signature d'une application. Un certificat peut être auto-signé.

## Attaques TLS diverses

### *L'attaque par renégociation (2009)*

En Novembre 2009, Marsh Ray et Steve Dispensa découvrent (*Marsh Ray et Steeve Dispensa, phonephactor.com, Renegotiating TLS, nov. 2009, [http://www.phonephactor.com/sslgapdocs/Renegotiating\\_TLS.pdf](http://www.phonephactor.com/sslgapdocs/Renegotiating_TLS.pdf)*) une faille dans le protocole SSL/TLS qu'ils nomment *Authentication Gap*. Cette faiblesse provient du clivage logiciel des protocoles HTTP et TLS. Lors d'une requête HTTPS ordinaire (sans certificat client), le serveur déchiffre l'entête HTTP et découvre que le fichier requis exige un certificat client. Il initialise alors une deuxième session TLS avec mutuelle authentification. L'attaquant se place en MIM (*Man In the Middle*) et détourne la première requête HTTPS sans certificat. Il injecte alors sa propre requête HTTP (codée de telle manière à annuler celle qui sera produite ultérieurement par le client légitime. Par la suite il se comporte en un relais passif transportant la deuxième session TLS, avec le certificat client.

### *L'attaque BEAST (2011)*

En septembre 2011 Juliano Rizzo and Thai Duong ont publié lors de la conférence *ekoparty* à Buenos Aires une attaque dénommée "*Browser Exploit Against SSL/TLS*" (BEAST). Cette dernière permet de retrouver par comparaison (Oracle) le contenu chiffré d'un entête HTTP, comme par exemple un cookie (utilisé par une session Paypal dans leur article). L'attaque s'appuie sur une propriété d'un chiffrement en mode CBC (Cipher Block Chaining), lorsque la première valeur ( $IV_0$ ) est connue; elle a été analysée en 2006 par Gregory V. Bard (Gregory V. Bard "Challenging but feasible Block wise-Adaptive Chosen-Plaintext Attack On SSL", 2006)

Soit  $E$  un algorithme de chiffrement symétrique (par exemple 3xDES)

$IV_0$  connu



$$C_0 = E \{ IV_0 \text{ exor } M_0 \}, IV_1 = C_0$$

$$C_1 = E \{ IV_1 \text{ exor } M_1 \}, IV_2 = C_1$$

$$C_k = E \{ IV_k \text{ exor } M_k \}, IV_{k+1} = C_k$$

La suite des valeurs  $C_i$  représente les éléments chiffrés obtenus à partir des blocs en clair  $M_i$ . Dans le cas du 3xDES la taille de ces blocs est de 8 octets, soit 64 bits d'entropie ( $2^{64}$ ). L'attaquant connaît un prochain  $IV_{i+1}$  par observation des informations préalablement chiffrées. Il injecte la donnée  $IV_{i+1} \text{ exor } IV_j \text{ exor } M^*$  ( $IV_j$  un IV préalablement observé ou connu) dans le but de deviner un bloc précédant en clair  $M_j$ , vérifiant  $C_j = E ( IV_j \text{ exor } M_j )$ , d'où l'on déduit

$$C_{i+1} = E ( IV_{i+1} \text{ exor } IV_{i+1} \text{ exor } IV_j \text{ exor } M^* ) = E( IV_j \text{ exor } M^* ),$$

qui est égal à  $E(IV_j \text{ exor } M_j)$  pour  $M^*=M_j$

Pratiquement l'attaque exige l'observation des valeurs chiffrées sur le réseau (pour collecter les valeurs  $C_k$ ) et l'injection de données en clair dans le navigateur (via un script dans l'attaque de 2011). La force brute implique cependant l'injection de  $2^{64}$  blocs dans le cas du 3xDES. L'insertion d'octets de bourrage dans l'entête HTTP (de 0 à 7) permet de réduire le nombre d'essais.

Par exemple  $M^* = B7 B6 B5 V7 V6 V5 V4 X$ , permet de retrouver en 256 essais la valeur de l'octet  $X$  (de rang 3) les octets de préfixe  $B$  assurant le bourrage et ceux de préfixe  $V$  étant connus.

### L'attaque Lucky Thirteen (2013)

En 2013 Nadhem J. AlFardan et Kenneth G. Paterson, ont publié une attaque nommée Lucky Thirteen ("Lucky Thirteen: Breaking the TLS and DTLS Record Protocols" Nadhem J. AlFardan and Kenneth G. Paterson, 2013).

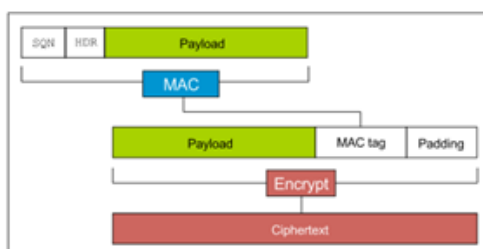


Figure 1: D(TLS) encryption process

Cette attaque s'appuie sur un défaut de sécurité des protocoles TLS et DTLS: l'intégrité des octets de bourrage (padding) utilisés pour le chiffrement en mode bloc, n'est pas garantie par un HMAC.

Selon les standards TLS et DTLS la structure de *padding* comprend un champ longueur (un octet L) suivi de L octets dont la valeur est fixée à L. Par exemple 0x00, 0x01 0x01, 0x02 0x02 0x02, et 0xFF suivi de 255 valeurs à 0xFF.

Un message DTLS comporte un champ compteur (SQN, 8 octets) et un entête (HDR, 5 octets). L'ensemble SQN || HDR possède donc une longueur de 13 octets.

L'attaque Lucky Thirteen est du type "*Distinguishing attack*", c'est-à-dire que l'attaquant est capable de distinguer le chiffrement de deux messages M0 et M1.

Considérons deux charges DTLS, Payload0 (P0) et Payload1 (P1)

$P0 = 32 \text{ octets} + 256 \times 0xFF$  (soit 256 octets de padding)

$P1 = 287 \text{ octets} + 0x00$  (soit un octet de padding)

Soit 18 blocs de 16 octets (288 octets), chiffrés par l'algorithme AES (en mode CBC 128 bits)

Les messages DTLS M0 et M1 transportant ces charges (P0, P1) comprennent le préfixe SQN || HDR et le suffixe T || Padding. Soit

$M_0 = \text{SQN} || \text{HDR} || P_0 || T || \text{pad}$

$M_1 = \text{SQN} || \text{HDR} || P_1 || T || \text{pad}$

L'attaquant observe

$M = \text{SQN} || \text{HDR} || P_i || T || \text{pad}$

Il forge

$M' = \text{SQN} || \text{HDR} || P_i$ , ce message est traité par le protocole DTLS

Si la charge est égale à  $P_0$ , le message  $M'$  est interprété après déchiffrement comme 12 octets de payload + 20 octets de HMAC-SHA1 et 256 octets de padding.

Si la charge est égale à  $P_1$ , le message  $M'$  est interprété après déchiffrement comme 267 octets de payload + 20 octets HMAC-SHA1 + 1 octet de padding.

Les temps de calcul du HMAC sont en conséquence différents.

L'attaquant souhaite obtenir la valeur en clair ( $P_4$ ) d'un bloc chiffré de 16 octets ( $C_4$ ), il a également observé la valeur  $C'$  utilisée lors d'un précédent chiffrement en mode CBC (dans la séquence  $C' || C_4$ )

$C_4 = E(P_4 \text{ exor } C')$  ( $E$ =chiffrement)

L'attaquant forge une valeur  $C'$  exor  $X$  ( $X$  une valeur comprise entre 0 et FF), puis un message comportant 4 blocs de 16 octets ( $C_1, C_2, C' \text{ exor } X, C_4$ )

$M = \text{SQN} || \text{HDR} || C_1 || C_2 || (C' \text{ exor } X) || C_4$  (soit  $64+13=77$  octets)

Le protocole DTLS déchiffre la valeur ( $D$ = déchiffrement)

$P_4' = D(C_4) \text{ exor } (C' \text{ exor } X) = (P_4 \text{ exor } C') \text{ exor } C' \text{ exor } X = P_4 \text{ exor } X$

Si le dernier octet de  $P_4'$  est 0, le protocole DTLS interprète  $M$  comme un message de de 13 + 43 octets (soit 56 octets) suivi d'un HMAC de 20 octets et d'un octet de padding ( $4 \times 16 = 64 = 43 + 20 + 1$ ). Le calcul du HMAC s'applique donc à 56 octets

Si le dernier octet de  $P_4'$  indique une longueur de padding dont la structure est correcte (par exemple 0x01 0x01, la suite la plus probable) le calcul du HMAC s'applique à 55 octets au plus. L'algorithme HMAC utilise deux blocs de 64 octets pour une longueur inférieure à 55 et au moins trois

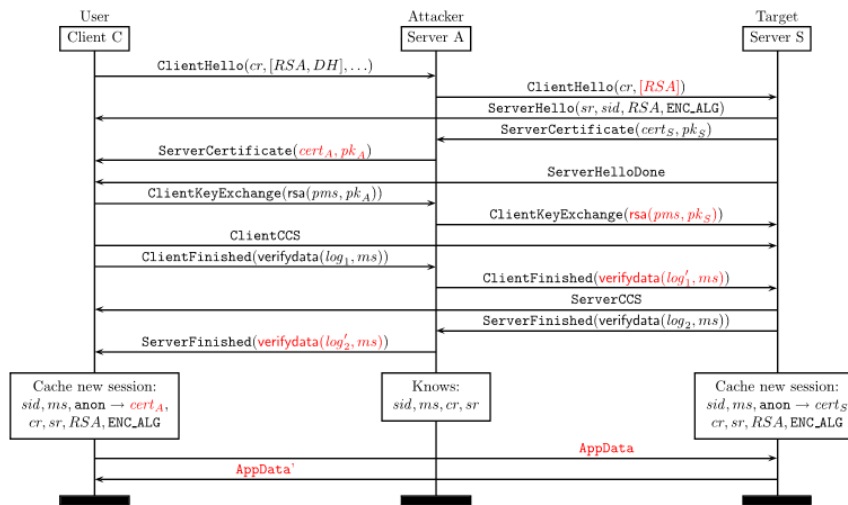
blocs de 64 octets dans le cas contraire. Cette différence provient de l'ajout de 8 octets (indiquant la taille originale) par l'algorithme SHA1 au message à traiter.

Si la structure du padding est incorrecte, le message est considéré sans padding, le calcul du HMAC s'applique à 57-octets ( $57 = 77 - 20$ )

Dans tous les cas de calcul HMAC détecte une intégrité erronée, et un message d'erreur est généré. Le cas d'un padding de longueur supérieur à un, implique un temps calcul plus court.

### Attaque TLS Triple handshake

Cette attaque<sup>17</sup> a été publiée le 4 mars 2014 par une équipe de recherche de l'INRIA Rocquencourt. Elle est basée sur un serveur MIM malveillant. Le principe de l'attaque est de forger un pre-mastersecret (PMS) identique pour un serveur MIM malveillant et un serveur licite lors d'une session TLS full. Par la suite le serveur MIM est capable de déchiffrer les messages des sessions en mode resume



### HeartBleed 2014

HeartBleed est un bug du logiciel OPENSSL détecté en 2014, qui permet de lire un bloc aléatoire de mémoire de 64ko.

<sup>17</sup> <https://secure-resumption.com/>

Heartbeat (RFC 6520) est un protocole transporté par la couche *RecordLayer* de TLS, dont il teste le bon fonctionnement grâce à un mécanisme d'écho. Il existe deux types de messages, requête et réponse. Le premier écrit un bloc de données en mémoire, le deuxième lit le bloc précédemment écrit.

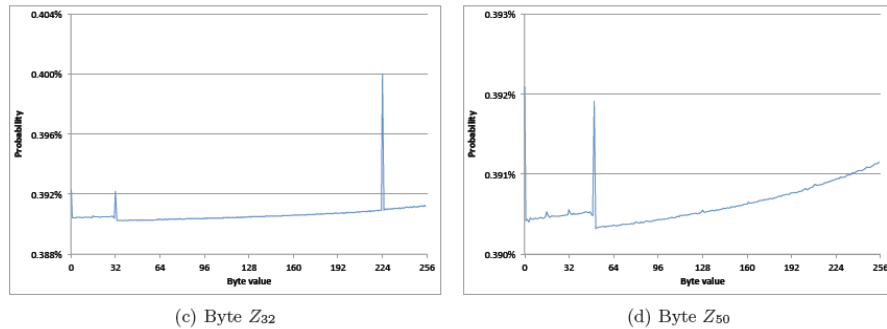
```
struct {
    HeartbeatMessageType type;
    uint16 payload_length;
    opaque payload[HeartbeatMessage.payload_length];
    opaque padding[padding_length];
} HeartbeatMessage;
```

Le bug provient du fait que la taille du bloc d'une requête (payload\_length) n'est pas vérifiée. La lecture associée (dont la taille de bloc est identique) permet d'obtenir une zone mémoire aléatoire (jusqu'à 64 Ko)

#### ***L'attaque RC4 du L'attaque Royal Holloway (2013)***

RC4 est l'algorithme le plus utilisé par le protocole TLS. L'attaque *Royal Holloway*, publiée en juillet 2013 par un groupe de chercheurs ("On the Security of RC4 in TLS and WPA", Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, Jacob C. N. Schuldt) repose sur l'existence de polarisations ("biais") simple octet ou double octet observés pour l'algorithme RC4.

Une polarisation simple octet est la probabilité d'obtenir un octet Ksi au rang  $i$ , avec  $i \geq 1$ . Pour un générateur pseudo aléatoire idéal cette probabilité est de  $1/256$  (0,390625 %). Cette probabilité a été estimée de manière expérimentale à l'aide d'une distribution  $2^{24}$  keystreams Ksi utilisant des clés de 128 bits. Une polarisation significative est observée pour tous les octets Ksi pour des rangs inférieurs à 256. Par exemple le 32<sup>ième</sup> octet a une probabilité de 0,4% d'avoir pour valeur 224



L'attaque nécessite le chiffrement d'un message identique par un nombre  $2^n$  de sessions utilisant des clés de 128 bits différentes.

$2^{26}$  sessions permettent de retrouver les 80 premiers octets avec une probabilité d'au moins 50 %.

$2^{32}$  sessions permettent de retrouver les 256 premiers octets avec une probabilité d'au moins 96%.

Une polarisation double octet est la probabilité d'obtenir une valeur  $K_{si+1}$  au rang  $i+1$  connaissant la valeur  $K_{si}$  au rang  $i$ . Dans le cas d'un générateur pseudo aléatoire idéal, cette probabilité est de  $1/256$ .

L'attaque nécessite le chiffrement d'un message identique par un nombre  $2^n$  de sessions utilisant des clés de 128 bits différentes. Le premier octet du message en clair est connu, le but de l'attaquant est d'obtenir la valeur des 16 octets suivants. Au-delà de  $11.2^{30}$  sessions le message en clair est récupéré avec une probabilité de 99%.