Introduction
à la Cyber Sécurité - 2017

Machine Enigma, *Imperial War Museum*, Londres

Pascal Urien - Telecom ParisTech

Introduction 5
Ordinateurs et sécurité des applications 5
Le Réseau Téléphonique Commuté (RTC) 7
L'internet 7
L'internet sans fil, les réseaux radio 8
Transfert de fichiers et streaming 9
Roaming, VPN 9
Du Data Center au Cloud 9
L'Architecture Logicielle Réseau selon REST 10
Sécurité du Cloud 10
Modèle du Cloud - NIST Special Publication 800-145 11
Surface d'attaque du Cloud 12
Eléments de sécurité du Cloud 12
Network Isolation and Segmentation 13
Application Isolation 13
Data Security 13
Identity and Access Management (IAM). 13
L'émergence du monde connecté et l'internet des objets 14
Protocoles de l'Internet of Things (IoT) 15
CoAP (Constrained Application Protocol) 15
LWM2M (Lightweight Machine to Machine Technical Specification) 15
MQTT (MQ Telemetry Transport) 15
Quelques plateformes logicielles pour l'IoT 16
Thread 17
Open Connectivity Foundation OCF 19
HAP 20
Hue Smart Lighting 20
Brillo et Weave 21
Sécurité des Systèmes Industriels 21
SCADA 21
STUXNET, une arme logicielle 22
Les Voitures Connectées. 23
Attaques récentes sur les voitures connectées 24
L'émergence des Crypto-Monnaies, Blockchain, Smart Contracts 25
Bitcoin 25
Ethereum et smart contract 26
Sécurité des Applications Distribuées, Emergence de la Cyber Sécurité 27
Sécurité des applications 27
Sécurité des protocoles de communication et des réseaux 28
Sécurité du système d'exploitation 28
Sécurité Hardware 29
Quelques paradigmes de sécurité 29
Classification des types d'attaques 29

Heuristiques de défense: placebo, vaccin, défense immunitaire 30
Facteurs de Vulnérabilité: Complexité, Extensibilité, Connectivité 30
Une Méthodologie de Cyber Attaque: Empreinte, Collecte, Inventaire..... 31
Systèmes Embarqués 31
Principes de sécurité. 32
Identification 32
Authentification 32
Confidentialité 32
Intégrité 32
Non-répudiation 33
Disponibilité 33
Au sujet de la confiance 33
Sécurité des Réseaux 33
Sécurité au niveau physique 35
Sécurité au niveau MAC 35
Sécurité au niveau réseau/transport 35
Couche de Sécurité entre transport et application 35
Sécurité au niveau application 36
Eléments de Sécurité 36
Authentification - Autorisation 36
Identité et Organisations 38
Taxonomie des méthodes d'authentification 39
Mécanismes symétriques: mot de passe, pre-shared key, provisionning 39
Mécanismes Asymétriques 41
Les tunnels 41
Les faiblesses du protocole IP 42
Les solutions sécurisées IP classiques: VPN, TLS, Kerberos, SSH 43
IPSEC 43
TLS/SSL 45
Au sujet du Phishing 47
Kerberos 47
PPTP-EAP 48
SSH 49
Limitation des protections offertes par les pare-feu. 50
Au sujet des normes Critères Communs (CC) 51
Heuristiques d'Attaque 53
L'intrusion. 53
Programmes Malveillants, Virus, Vers 53
Exemple le ver Blaster (2003) 55
Au sujet des botnets 55
Au sujet des rootkits 56
Le buffer overflow 56
Le Fuzzing 58
L'injection SQL 58
Le Cross Site Scripting CSS 58

Cross Site Request Forgery CSRF 58
Autour de la Sécurité des Systèmes d'Exploitation 58
Stratégie de défense du système Windows (2005) 58
Principes de sécurité du système Android 59
Intégrité du code, obfuscation, TPM 60
Exemple de classification des attaques pour les systèmes embarqués. 61
White-Box Cryptography (WBC) 62
Les Canaux Cachés. 63
Single Power Analysis 63
Differential Power Analysis 63
Faites moi confiance ? 20 ans de bugs et heuristiques 65
Attaques sur les fonctions de hash MD5 et SHA-1 71
Quelques TOP10 d'attaques 72
Le TOP9 des menaces visant le cloud computing en 2013 72
OWASP TOP10 2013 73
Attaques TLS diverses 75
L'attaque par renégociation (2009) 75
L'attaque BEAST (2011) 76
L'attaque Lucky Thirteen (2013) 77
Attaque TLS Triple Handshake (2014) 79
HeartBleed (2014) 80
L'attaque RC4 du L'attaque Royal Holloway (2013) 80

Introduction à la Cyber Sécurité

Introduction

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

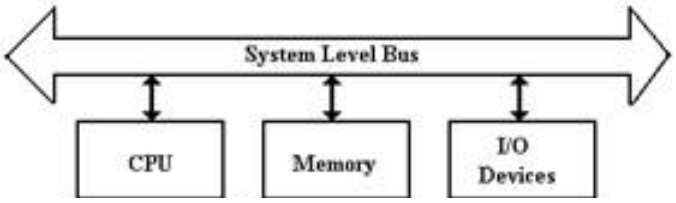
Ordinateurs et sécurité des applications

Dans un premier temps[1] 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

[1] Un des premiers ordinateur, "Colossus", a été construit en 1944 par l'équipe de Tommy Flowers selon les spécifications de Max Newman. Il comportait dans sa dernière version, 2400 tubes à vide.

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 transistor a été inventé fin 1947 par John Bardeen, William Shockley et Walter Brattain (prix Nobel de physique en 1956). L'entreprise INTEL fut fondé en 1968 par les docteurs Gordon Moore, Robert Noyce et Andrew Grove; elle créa en 1971 le 1ier microprocesseur 4 bits (le 4004, comportant 2 300 transistors). Le PDP-11 (Programmable Data Processor) commercialisé par Digital Equipment Corporation (DEC) entre 1970 et 1993 a popularisé l'architecture classique des ordinateurs comprenant, un CPU, un bus système, des mémoires, et des dispositifs d'entrée/sortie (I/O).



PDP 11, 1970

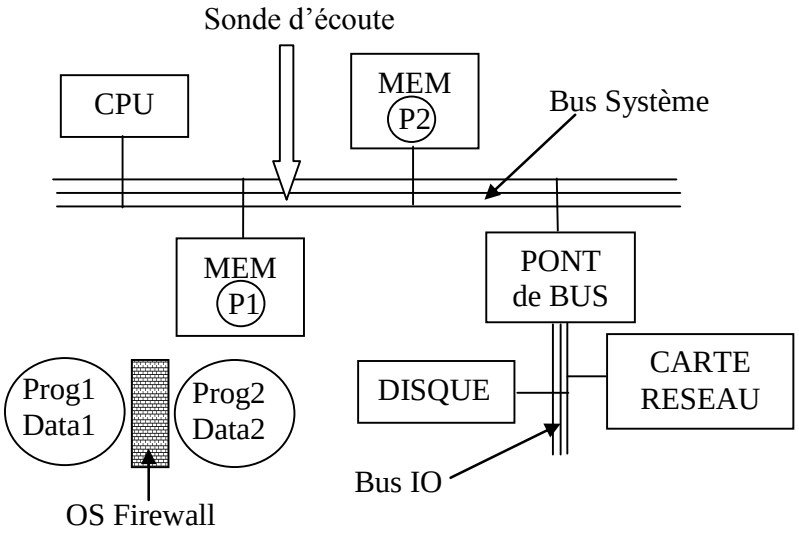
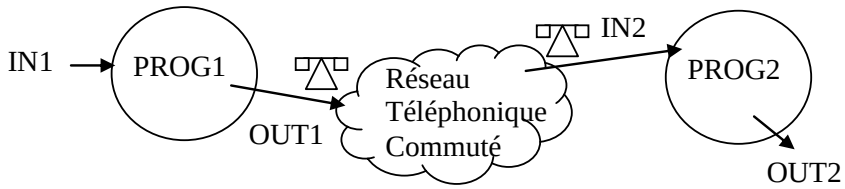


Illustration des attaques et défenses sur un système informatique.

Le système d'exploitation CP/M (et ancêtre du DOS), créé en 1974 par *Gary Kildall*, fondateur de *Digital Research* utilisait des lignes de commande selon la syntaxe *ProgramName FileIn FileOut*.

Le Réseau Téléphonique Commuté (RTC)

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



L'internet

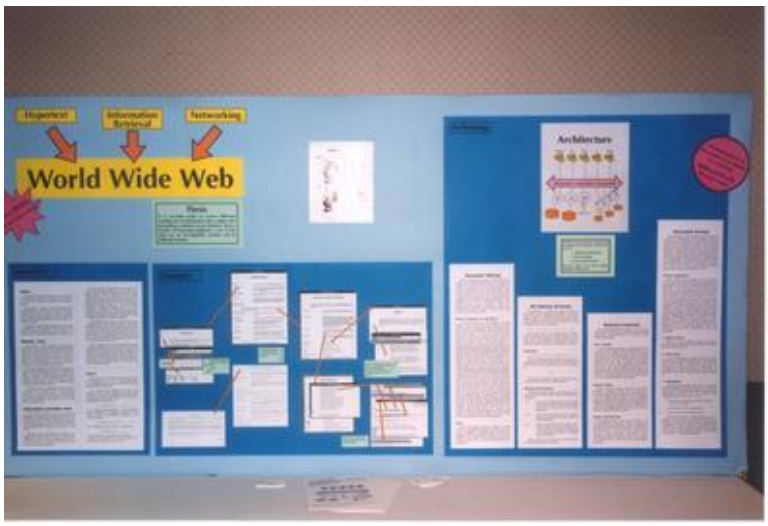
Enfin l'émergence de l'internet[2], puis de la toile d'araignée mondiale (WEB[3]) a permis de concevoir des systèmes distribués à l'échelle planétaire, les composants logiciels sont répartis sur des systèmes informatiques hétéroclites, le réseau n'est pas sûr, le nombre d'utilisateurs est important.

D'abord absente des premières spécifications de l'Internet (*Security Architecture for the Internet Protocol*, RFC 825, 1995) puis du WEB (*Secure Socket Layer*, SSL 3.0, 1996), 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 au piratage informatique ou à l'espionnage.

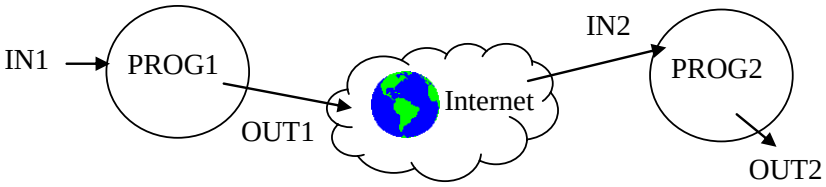
---

[2] Vinton G. Cerf and Roberty E. Kahn "A Protocol for Packet Network Intercommunication", IEEE Trans on Comms, Vol Com-22, No 5 May 1974

[3] Tim Berners-Lee. "The World Wide Web", Poster, ACM Hypertext 91 Conference.



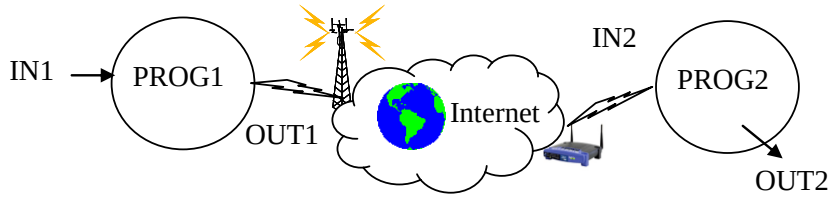
Poster de Tim Berners-Lee à la conférence ACM Hypertext 91.



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[4] ou WiMAX[5] ou réseaux cellulaires 3G/4G/5G. 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 et la confidentialité des données.

[4] IEEE 802.11b, Septembre 1999
[5] IEEE 802.16e, Décembre 2005



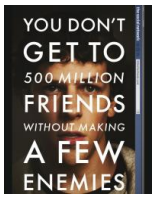
Transfert de fichiers et streaming

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 (streaming)*. 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*). Chez les particuliers une box (modem ADSL) établit un lien point à point avec le réseau de l'*Internet Service Provider* (ISP).

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.

L'Architecture Logicielle Réseau 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 une 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.

Sécurité du Cloud

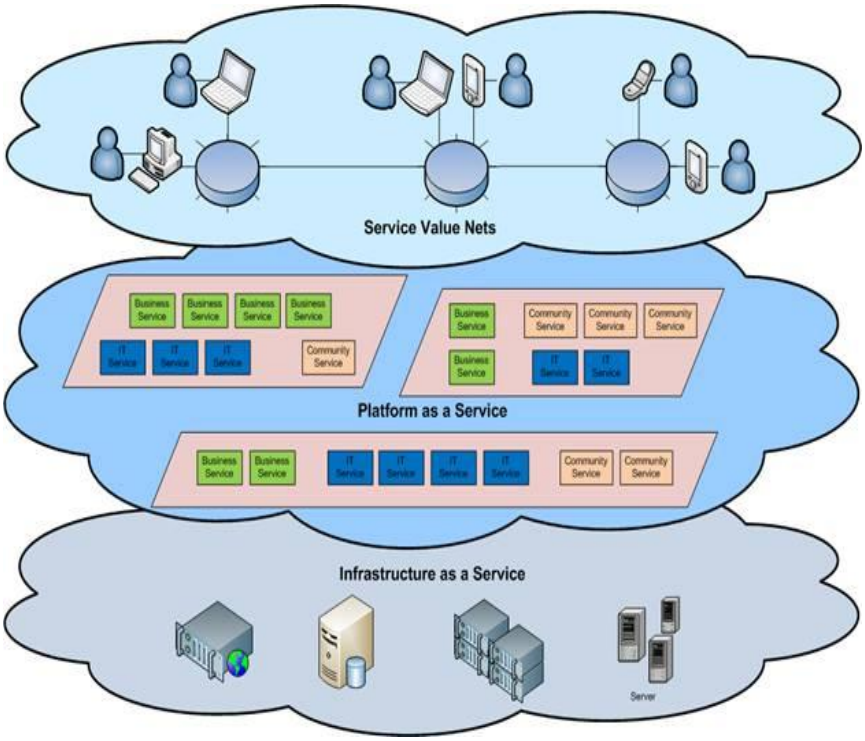
Selon la définition du NIST (*Special Publication 800-145, 2011*), le *cloud computing* est un modèle qui permet un accès efficace, omniprésent, et à la demande, à un ensemble (*pool*) partagé de ressources informatiques configurables (telles que réseaux, serveurs, stockage, applications et

services). Ces ressources peuvent être provisionnées rapidement et libérées avec un effort de gestion minimal et une interaction réduite avec le fournisseur de services. Il est important de remarquer que l'infrastructure est la propriété du fournisseur de service (service provider).

Modèle du Cloud - NIST Special Publication 800-145

Ce modèle est composé de cinq caractéristiques essentielles, trois modèles de service , et quatre modèles de déploiement .

Les cinq caractéristiques essentielles sont: 1) la provision de ressources à la demande et en self service, 2) l'accès aux ressources via le réseau, à partir de clients légers, 3) le partage de ressources multi utilisateurs et géographiquement distribuées, 4) l'élasticité des ressources disponibles, 5) la mesure de la consommation des ressources.



Une plateforme de *Cloud Computing* est divisée en trois modèles de services :

- 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. Le client accède uniquement à la configuration des applications.

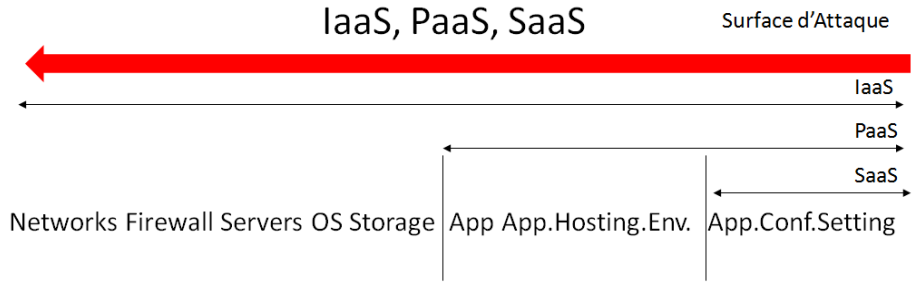
- PaaS ou *Plateform as a Service*. C'est un environnement (APIs, Machine Virtuelles[6]) permettant de construire/déployer des applications par exemple Google App Engine ou Microsoft Azure.

- 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.

Les quatre modèles de déploiement du cloud comportent 1) le cloud privé, 2) le cloud communautaire, 3)le cloud public, 4) le cloud hybride.

Surface d'attaque du Cloud

La surface d'attaque augmente selon l'axe SaaS, PaaS, IaaS, puisque le nombre de ressources mises à disposition de l'utilisateur augmente.



Eléments de sécurité du Cloud

Outre les procédures et normes de qualité, d'audit, et de conformité liées aux locaux, aux contrôles d'accès physique, aux traitements d'incidents de sécurité, aux incendies et dégâts des eaux, les principaux domaines de sécurité de l'infrastructure du *cloud* sont les suivants.

[6] L'hyperviseur Xen fut créé à la fin des années 90, par Keir Fraser et Ian Pratt, dans le cadre du projet de recherche *Xenoserver* de l'université anglaise de Cambridge.

Network Isolation and Segmentation

1) *Network Isolation and Segmentation*. Ces mesures visent la protection de l'infrastructure par isolation/segmentation et analyse des flux. Elles s'appuient sur les éléments suivants:

– Firewalls réseau, et firewalls au niveau VM utilisateur,

– Dispositifs IDS (*Intrusion Detection System*), IPS (*Intrusion Prevention System*), DLP (*Data Leak/Lost Prevention*),

– Logs du système,

–Détection de clients malveillants,

–Chiffrement des informations échangées entre *data centers*, mais respect des lois dans le pays d'accueil (*Patriot Act* par exemple)

Application Isolation

2) *Application Isolation*. Ces mesures adressent l'isolation logique des applications hébergées. Elles s'appuient sur les éléments suivants:

– La Virtualisation

– Les bugs de sécurité des hyperviseurs,

– Les canaux cachés (side channel) inter VM

- Le matériel sécurisé, hyperviseur certifié (Critères Commun - CC), Unix durçi.

Data Security

3) *Data Security*. Ces mesures adressent la confidentialité des données échangées et stockées.

– Protocoles de sécurité réseau TLS, IPSEC, VPN, SSH

– Chiffrement des fichiers

- Protection, chiffrement des bases de données *Multi Tenant*.

- Gestion des clés clients sécurisée par des HSM (*Hardware Security Module*)

Identity and Access Management (IAM).

4) *Identity and Access Management (IAM).* Ces mesures adressent le contrôle d'accès à l'infrastructure du cloud.

– Login password,

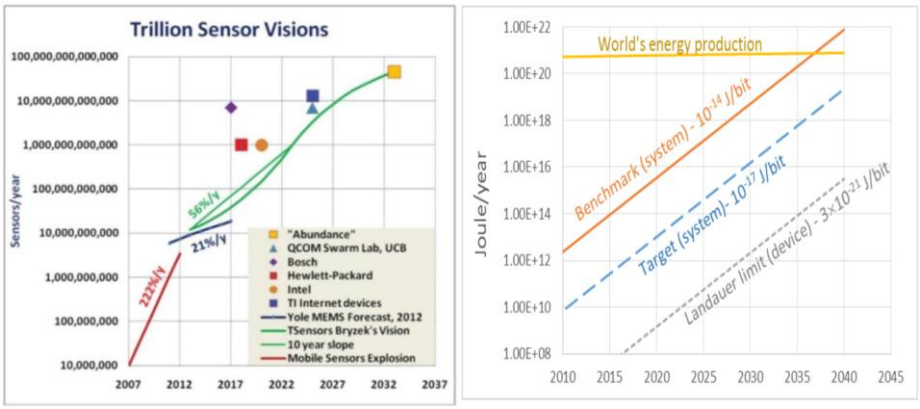
- One Time Password (OTP),

– Certificats,

– Stockage sécurisé de clés,

–Dispositifs *Hardware Secure Module* (HSM).

L'émergence du monde connecté et l'internet des objets



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, Intelligent Transport System -ITS-*), les objets connectés (lunettes, montres, 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 objets (*Internet of Things* IoT).

Selon un white paper[7] de CISCO le nombre d'objets connectés était de 10 milliards en 2010 et pourrait atteindre 50 milliards en 2020. L'article[8] évoque un nombre de 100,000 milliards de capteurs à l'horizon 2035, et l'émergence des *Cyber Physical System* (CPS)

7 "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

8 Rebooting the IT Revolution: A Call to Action" (SIA/SRC), 2015

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.

Protocoles de l'Internet of Things (IoT)

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

- Classe C0, RAM << 10KB, code << 100KB, ces systèmes (capteurs) n'ont pas les ressources nécessaires pour gérer des connections sécurisées.

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

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

CoAP (Constrained Application Protocol)

Le protocole CoAP (*Constrained Application Protocol*, RFC 7252, 2014) sécurise (de manière optionnelle) les échanges d'information de capteurs en utilisant DTLS (TLS avec un transport UDP) ou TLS.

LWM2M (Lightweight Machine to Machine Technical Specification)

LWM2M (*Lightweight Machine to Machine Technical Specification*) est un environnement basé sur CoAP qui gère la communication entre des objets logés dans des clients LWM2M et des serveurs LWM2M. Il comporte deux modes de transport (UDP/IP et SMS) ainsi que quatre interfaces fonctionnelles, 1) le bootstrap, 2) l'enregistrement des clients avec les serveurs, 3) l'administration des objets, 4) les rapports d'information.

MQTT (MQ Telemetry Transport)

MQTT[9] (MQ Telemetry Transport) d'origine IBM, est un protocole de type *Publish-Subscribe* (bus asynchrone) sécurisé par TLS. Les nœuds *publisher* délivrent de manière asynchrone de l'information aux serveurs *broker de messages*, les nœuds *subscriber* collectent ces informations.

[9] IBM, Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry, 2012

Quelques plateformes logicielles pour l'IoT

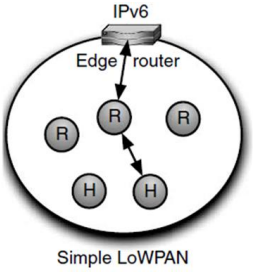
L'internet des objets combine un système informatique de type embarqué et une interface de communication telle que Wi-Fi (IEEE 802.11) ou ZigBee (IEEE 802.15.4).

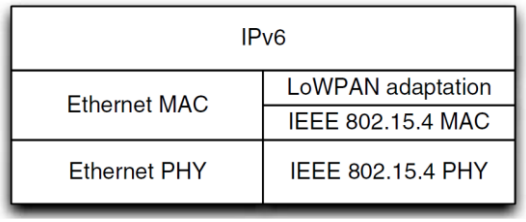
Un objet se décompose généralement en quatre entités logiques et matérielles: 1) un circuit électronique (board) qui comporte typiquement un processeur, 2) un SOC (*System On Chip*) radio, 3) un système d'exploitation ou un bootloader, 4) une pile de communication et un environnement (*framework applicatif*). La syntaxe des messages applicatifs s'appuie sur JSON (*JavaScript Object Notation*), un format d'échange de données (*Data Interchange Format*) en mode texte. La structure d'un document JSON est décrit par un schéma JSON. Les messages applicatifs sont échangés selon le paradigme REST, à l'aide des protocoles HTTP ou CoAP.

Il existe de nombreux systèmes d'exploitation, importés de l'informatique traditionnelle, ou adaptés pour des systèmes embarqués avec des ressources de traitement (*processing*) ou de mémoire modeste, par exemple Linux, Contiki, Riot, Iotivity, AllJoyn, Brillo, mbed OS...

Du point de vue sécurité, on constate trois grandes approches; 1) la sécurité au niveau MAC (Wi-Fi, ZigBee) par exemple pour les ampoules connectées *Philips Hue Bulbs*, 2) la sécurité à l'aide des classiques protocoles TLS/DTLS (piles logicielles Thread ou OCF), et enfin 3) la sécurité au niveau de l'application (par exemple l'architecture HomeKit Accessory Protocol - HAP- de la société Apple).

De surcroit des mécanisme de contrôle d'accès utilisant des ACLs (*Access Control List*) ou des jetons (*OAuth 2.0 Authentication*), peuvent être appliqués dans la couche applicative.



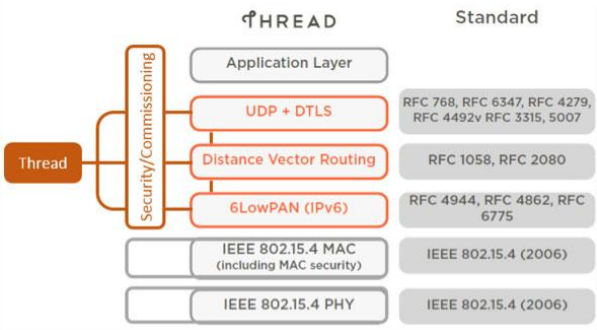


Le transport de paquets IPv6 natifs, dont le MTU (*Maximum Transmission Unit*, taille de paquet minimale transmise sans fragmentation) est de 1280 octets, n'est pas compatible avec la taille maximale de 127 octets des trames

IEEE 802.15.4 (les entêtes IPv6 et TCP occupent au moins 40+20 octets). Le standard 6LoWPAN[10] (*IPv6 Low power Wireless Personal Area Networks*) finalisé en 2007 (RFC 4919) définit une couche d'adaptation (*adaptative layer*) qui réalise des opérations de segmentation/assemblage et également de mode de routage, internes à la couche d'adaptation (*mesh-under*) ou implémentée dans la couche IPv6 (*route-over*). Un réseau 6LoWPAN comprend des nœuds ordinaires (*Host*), des routeurs (*Routers*) et un routeur de bord (*Edge Router*) connecté au réseau IPv6. L'adresse IPv6 (128 bits) d'un nœud comporte un suffixe de 64 bit basé sur l'adresse MAC et un préfixe alloué selon le protocole *Neighbor Discovery Router Advertisement* (RA).

Le protocole IPSEC est bien sur compatible avec cet environnement. Cependant ZigBee dispose d'algorithmes de chiffrement de trame, typiquement selon le mode AES-CCM 128 bits, et possède un jeu de trois clés. Un secret à long terme (*Master Key*) partagé par tous les membres du réseau, permet au terme du protocole SKKE (*Symmetric-Key Key Establishment*) d'établir une clé de ligne (*Link Key*) pour les communications unicast. Cependant les réseaux ZigBee usuels partagent une même clé de ligne, distribuée par un nœud particulier le *Trust Center* (TC), grâce au MasterKey. Une clé réseau (*Network Key*) est commune à tous les participants et chiffre les trames émises en diffusion (broadcast).

Thread



Le consortium Thread définit une pile réseau s'appuyant sur 6LoWPAN et un routage de type route-over (Distance Vector Routing). La sécurité est délivrée par une couche DTLS munie du mécanisme d'authentification symétrique J-PAKE (Juggling Password-Authenticated Key Exchange), implémentant un mécanisme NIZK (Non-Interactive Zero-Knowledge), à

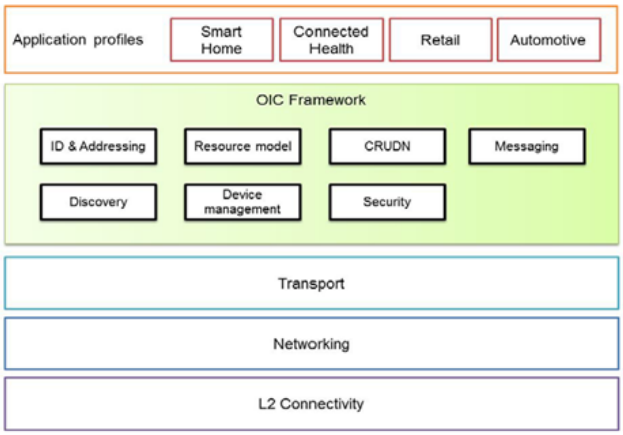
[10] 6LoWPAN: The Wireless Embedded Internet, Zach Shelby, Carsten Bormann, Wiley, 2009

l'aide d'échanges Diffie-Hellmann sur des courbes elliptiques, et de signature de Schnorr. Selon Thread un objet (*joiner*) s'identifie après d'un serveur d'authentification (*Commissioner*) localisé dans l'internet, selon un protocole nommé *Petitioning*. Cette opération est réalisée à l'aide d'une connexion point à point avec un commissioner, le routeur de bord (border router) ou un routeur. Le commissioner est un nœud IPv6 ou IPv4; dans ce dernier cas il est nécessaire de traiter les messages DTLS via un proxy. Au terme du processus de petitioning, le joiner récupère la clé de ligne ZigBee et d'autres information protégées par un mécanisme KEK (Key Establishment Key).

Thread est basé sur le système d'exploitation LINUX. Des sociétés telles que Silicon Labs (www.silabs.com) proposent des cartes mères de développement. Le thermostat NEST implémente les concepts Thread. L'article[11] met en évidence l'absence de Secure Boot et présente une attaque permettant de démarrer le système à partir d'une clé USB, puis de modifier ses fonctionnalités.

[11] "Smart Nest Thermostat: A Smart Spy in Your Home", Grant Hernandez, Orlando Arias, Daniel Buentello, and Yier Jin

Open Connectivity Foundation OCF



L'architecture définie par le consortium *Open Connectivity Foundation* (OCF) est similaire à Thread relativement au réseau. Elle s'appuie sur une pile 6LoWPAN sécurisée par le protocole DTLS. OCF introduit une entité applicative divisée en deux sous ensembles une couche (framework) définissant des éléments protocolaires (messages, sécurité, opérations CRUDN:Create, Read, Update, Delete, Notify...) et des profils métier tels que domotique, santé, automobile, commerce. L'association IoTivity conçoit des logiciels ouverts implémentant OCF et disponibles pour les systèmes d'exploitation Android, Tizen, Ubuntu , Windows, et Arduino!. Dans OCF le protocole DTLS définit l'identité de l'objet à l'aide de mécanismes cryptographiques (symétriques, pre-shared key) ou asymétriques (ECC, RSA, certificats...); de surcroit la couche applicative supporte un mécanisme de type *Access Control List* (ACL).

Un objet oic (Open Interconnect Consortium) logé dans un serveur est identifié par un chemin (oic://server/path). Il est associé à une requête (?query) adressant diverses propriétés (*property*, par exemple *Type* "rt", *Interface* "if", *name* "n", *Identity* "id") définies par des tuples *key=value*.

oic://server/path?key1=value1;key2=value2.

Ainsi la propriété interface ("oic.if.a") fixe les opérations CRUDN (CREATE, RETREIVE, UPDATE) disponibles pour les *actuators* (oic://server/path?if="oic.if.a"). Les ressources d'un l'objet sont également associées à une liste de propriétés (tuples clé-valeur, par exemple {"settemp": 10} en notation JSON) sur lesquelles s'appliquent des procédures CRUDN.

L'URI oic://server/path?key1=value1;key2=value2 est adapté à des protocoles de transport tels que HTTP ou CoAP.

HAP

Le modèle HAP (*HomeKit Accessory Protocol*) de la société APPLE s'appuie sur des WLAN Wi-Fi ou BLE (*Bluetooth Low Energy*), avec une couche applicative basée respectivement sur les protocoles HTTP ou GATT/ATT/L2CAP. Le paradigme de sécurité est symétrique. Les objets sont associés à un code PIN de 8 digits, et authentifiés via le protocole *Secure Remote Password procedure* (SRP, RFC 5054). Par la suite un objet HAP génère une paire de clés asymétriques[12] basée sur la courbe elliptique Ed25519, mises en œuvre pour l'authentification et la création des clés de session. Le chiffrement utilise l'algorithme ChaCha20-Poly1305.

Hue Smart Lighting

Les ampoules intelligentes Hue de Philips (*Hue Smart Lighting*) sont basées sur un réseau mesh zigBee, conforme aux spécifications ZLL (*ZigBee Light Link*). Toutes les ampoules partagent un clé maitre. Un contrôleur délivre aux nœuds du réseau une clé de ligne (*Link Key*) chiffrée à l'aide du *MasterSecret*. Ce dispositif intègre une interface radio et une interface *ethernet* munie d'une pile IPv4. Côté IP les commandes sont transportées en clair par des paquets UDP. L'article[13] présente une attaque par canaux caché de ce système qui met à profit l'absence de sécurité du contrôleur. Il introduit également une classification en 4 niveaux des attaques sur les objets; 1) Ignorer la fonction de l'objet et introduire un code malicieux, par exemple un botnet; 2) Réduire la fonction de l'objet, afin de créer un comportement défectueux; 3) Utiliser la fonction de l'objet de manière non pertinente; 4) Etendre la fonction de l'objet, par exemple en introduisant des canaux cachés.

On trouve dans l'article[14] de Colin O'Flynn un *reverse engineering* du contrôleur et des ampoules. Le contrôleur exécute un système d'exploitation linux, son processeur intègre des accélérateurs cryptographiques AES , 3xDES, MD5, SHA1. Les fichiers de mise à jour du système sont chiffrés par

---

12 iOS Security, iOS 9.0 or later, September 2015.
http://images.apple.com/euro/privacy/d/generic/docs/iOS_Security_Guide.pdf

13 Extended Functionality Attacks on IoT Devices: The Case of Smart Lights (Invited Paper), Eyal Ronen, Adi Shamir

14 A Light Bulb Worm, Details of the Philips Hue Smart Lighting Design, Colin O'Flynn - August 1, 2016.

l'algorithme AES dont la clé est stockée dans le processeur. De même les SOC zigbee du contrôleur et des ampoules intègrent un accélérateur cryptographique AES, utilisé pour le déchiffrement des fichiers de mise à jour. Cette analyse illustre la présence de mécanismes de sécurité matériels destinés à la protection des opérations de mise à jour des objets. Les clés cryptographiques et algorithmes associés sont protégés par des éléments hardware. L'article[15] analyse la sécurité du système Hue dont la principale faiblesse est l'usage de jetons d'autorisation intégrés aux requêtes HTTP (en clair), susceptibles d'être interceptés par des malware dédiés.

Brillo et Weave

La plateforme IoT *Weave* s'appuie sur le un système d'exploitation *Brillo* de Google, une version allégée d'Android qui comporte un noyau LINUX de 35Mo. La pile réseau supporte les WLANs IEEE 802.15.4, Wi-Fi, BLE. Elle utilise un paradigme REST transporté par les protocoles HTTPS ou XMPP, avec des jetons d'autorisation OAuth 2.0 délivré par des serveurs d'authentification (AS) Google.

Sécurité des Systèmes Industriels

SCADA

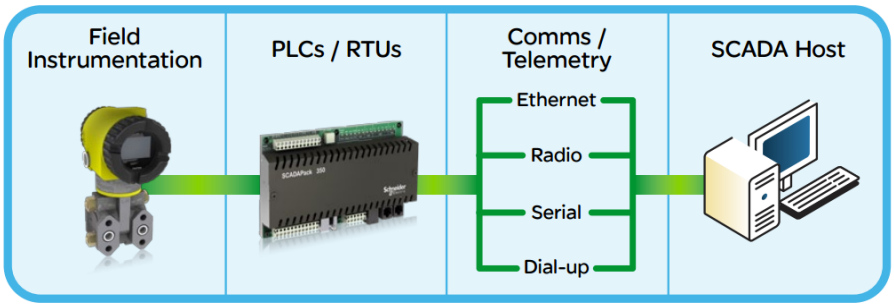
L'industrie, en particulier le contrôle des processus industriels (usines...) utilise également des technologies connectées. Par exemple le système de contrôle et d'acquisition de données SCADA[16] [17](*Supervisory Control And Data Acquisition*), comporte des dispositifs de contrôle tels que PLC (*programmable logic controller*) ou RTU (*remote telemetry unit*) communiquant par exemple via les protocoles MODBUS ou DNP3, fonctionnant en mode Master/Slave (sur des liaisons séries RS-232, RS-485) ou Client/Server (TCP/IP).

---

[15] Hacking Light bulbs: Security Evaluation of The Philips Hue Personal Wireless Lighting System, Nitesh Dhanjani, 2013

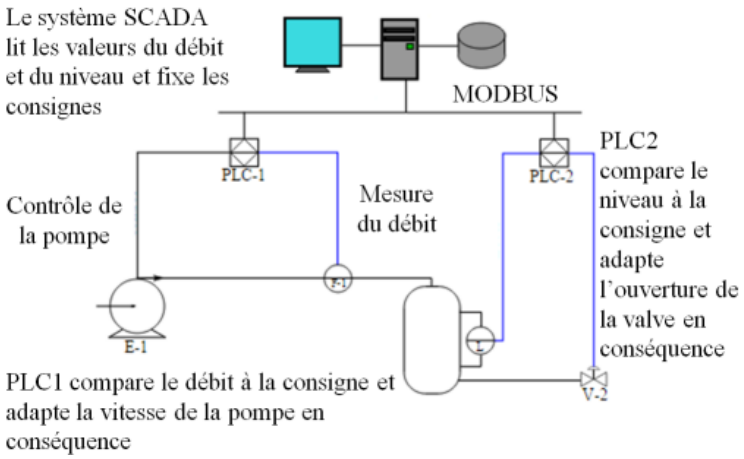
[16] http://www.schneider-electric.com/solutions/ww/fr/med/20340568/application/pdf/1485_se-whitepaper-letter-scadaoverview-v005.pdf

[17] Practical Modern SCADA Protocols:
DNP3, 60870.5 and Related Systems, ISBN 07506 7995



START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 octet	2 octets	2 octets	N octets	2 octets	CR LF

Structure d'un message MODBUS (pas d'éléments de sécurité)



Exemple de système SCADA.

STUXNET, une arme logicielle

Le vers informatique *Stuxnet* découvert en 2010 a permis[18] la destruction de machines industrielles (un millier de centrifugeuses...) en attaquant la

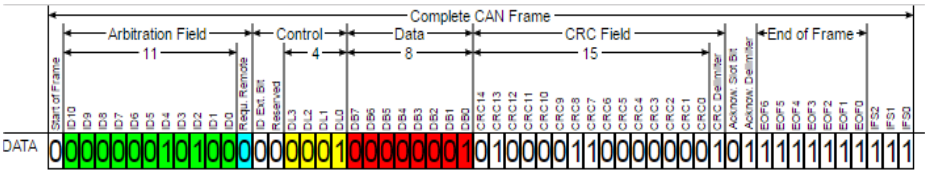
[18] Dissecting Stuxnet, Stanford University, https://www.youtube.com/watch?v=DDH4m6M-ZIU

plateforme SCADA WinCC/PCS7 de Siemens et en reprogrammant des PLCs (automates S7), depuis des machines *Windows* préalablement corrompues (des serveurs winCC). Le cœur de l'attaque s'appuie sur des défauts *Zero-Days* Windows et sur la mise à jour logicielle non sécurisée du PLC S7-300.

Les Voitures Connectées.

Une automobile moderne comporte un ensemble de boîtiers électroniques (une cinquantaine) dénommés *Electronic Control Units* (ECUs) connectés en réseau et destinés au contrôle et à la gestion du véhicule (freins, direction, pneus, ...).

Les ECU sont reliés à un ou plusieurs bus conformes au standard *Controller Area Network* (CAN), ISO 11898.



Un paquet CAN

Un paquet CAN comporte un identifiant (généralement 11bits), un champ longueur de données, des données (de 0 à 8 octets) et un CRC de 15 bits. Les paquets sont émis en diffusion sur les bus CAN, ils sont traités ou ignorés par les ECUs en fonction de la valeur de l'identifiant.

Des paquets de diagnostic sont utilisés uniquement à des fins maintenance.

Conformément aux normes ISO-TP l'entête des données (un ou deux octets) indique le type de trame (trame unique, première trame d'un bloc, trame d'un bloc, gestion de flux) et la longueur des informations.

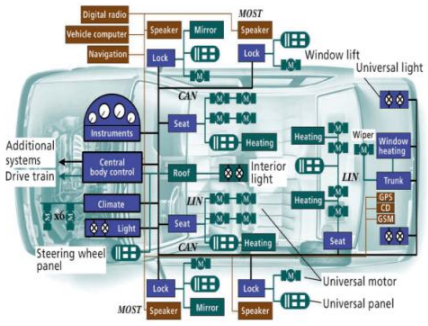
Le premier octet d'information est l'identifiant de service (*Service ID*) défini par le standard for ISO 14229. Par exemple *Security Access* (0x27) est une procédure de contrôle d'accès basée sur un mécanisme de défi/réponse utilisant un secret partagé (mot de passe...) utilisée pour les opérations de maintenance. Cependant il est important de remarquer que les échanges fonctionnels ne sont pas sécurisés.

Attaques récentes sur les voitures connectées

L'article[19] décrit des attaques par injection de paquets en mode maintenance ou fonctionnel; ces dernières permettent de contrôler entre autres, les freins, la direction, ou le régime moteur.

En 2015 l'article[20] décrit une attaque via le réseau cellulaire SPRINT visant une Jeep Cherokee. Le véhicule comporte un dispositif multimédia (auto radio, bluetooth, ...) nommé UCONNECT équipé d'un processeur Texas Instruments OMAP-DM3730, intégrant un système d'exploitation QNX. Cet équipement est connecté au bus CAN à l'aide d'un processeur Renesas V850; il possède également un port TCP 6667 ouvert sur le réseau cellulaire réalisant un service *D-Bus* destiné aux communications inter processus (IPC) et à l'appel de procédures distantes (RPC). L'exploit consiste à injecter un firmware modifié au coprocesseur V850 en exploitant une faille de type "buffer overflow". Par la suite il devient possible d'injecter à distance des paquets CAN à partir du port TCP 6667.

Les deux causes de cette attaque sont d'une part l'existence d'un *Buffer Overflow* et d'autre part un mécanisme de mise à jour logicielle non sécurisé.



De manière plus générale le rapport[21] du sénateur Américain *Edward J. Markey* met en évidence les failles de sécurité des points d'entrée des communications sans fil (*Wireless Entry Points*) dans les véhicules du marché, et les risques d'intrusion associés. "In a 2013 study that was funded by the Defense Advanced Research Projects Agency (DARPA), two researchers demonstrated their ability to connect a laptop to two different vehicles' computer systems using a cable, send commands to different ECUs through the CAN, and thereby control the engine, brakes, steering and other critical vehicle components."

---

19 Chris Valasek, Charlie Miller "Adventures in Automotive Networks and Control Units", 2014

20 Charlie Miller, Chris Valasek "Remote Exploitation of an Unaltered Passenger Vehicle", 2015

21 Senator Edward J. Markey, "Tracking & Hacking:Security & Privacy Gaps Put American Drivers at Risk", 2015

L'émergence des Crypto-Monnaies, Blockchain, Smart Contracts

Bitcoin

Les principes de la crypto monnaie Bitcoin[22] ont été définis en 2008 par l'article[23] écrit sous le pseudonyme de Satoshi Nakamoto. Cette monnaie numérique repose sur trois piliers, 1) l'émission de valeur basée sur un investissement en matériel informatique et sur la consommation d'énergie, 2) la gestion de transactions (transfert de valeurs) authentifiées par des signatures cryptographiques , 3) la publication des transactions dans des listes de blocs chainés (la blockchain) certifiés par les mineurs grâce à un mécanisme de *Proof of Work* (PoW), impliquant une dépense de temps CPU et d'énergie électrique.

Dans le monde bancaire classique Bob possède un compte. Lors d'une transaction par carte bancaire avec Alice, il est identifié/authentifié par sa carte, un cryptogramme est acheminé via le réseau des cartes bancaires vers sa banque qui délivre une autorisation.

Dans un contexte blockchain bitcoin Bob possède un identifiant (dénommé adresse, @bob) dérivé de sa clé publique (ECCC, Sepc256k1, g^x), et une clé privée (un nombre de 32 octets -x-). Les transactions sont publiques et enregistrées dans un livre de compte (le *ledger*) instancié par une série de blocs (la blockchain). Le crédit de @Bob et les transactions sont publiques et certifiés par la blockchain. Un avoir de @Bob est identifié par un montant et le hash (identifiant) d'une précédente transaction de versement, nommé UTXO (*Unspent Transaction Output*). Bob réalise un transfert d'UTXO au bénéfice d'Alice identifiée par son adresse @Alice; il peut attribuer un pourboire (f*ee*) au mineur déduit de l'UTXO. Il signe la transaction et la poste au système blockchain.

Toutes les 10 minutes les transactions sont assemblées dans des blocs (soit 144 blocs par jour environ). Un entête bloc (*header*) comporte entre autres le hash du bloc précédent, la racine d'un arbre de *Merkle* assemblé à partir des transactions, et un nombre aléatoire (*nonce*) qui réalise une solution à un problème difficile de hash.

Soit H la fonction H(x)=sha256((sha256)(x)).

Le mineur résout le problème :

---

22 Andreas M. Antonopoulos, "Mastering Bitcoin", O'REILLY, 2015

23 Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System.", 2008

H(nonce, header) < (65535 << 208) / difficulty

L'entropie du calcul est voisine de 32+log2(difficulty), soit 70 bits en 2016.

Un mineur utilise une machine spécialisée (le *rig*) pour accomplir cette opération. En 2016 le système ASIC AntMiner S9 réalise 14 Gh/s (14 milliards de calculs H(x) par seconde), pour un prix de 2400$, et une consommation électrique de 1,375 Watts, soit environ 0,1 Joule/Gh/s

Une difficulté de 2^{70} toutes les 10 minutes (#2^9 secondes) implique un hash rate d'environ 2.10^9 Gh/s (2^{61}). La consommation électrique journalière est de l'ordre de 4,800,000 KW/h pour un cout de 0,72 Million€ (avec un prix de 0,15€/KW/h). Le mineur reçoit une prime (*reward*) lorsqu'il gagne la compétition du minage, cette prime est de 12,5 BTC en 2017, soit 1800 BTC par jour, 1,8 Million€ pour un cours de 1000 $/BTC.

L'idée de création de monnaie du bitcoin s'inspire de l'image d'une mine d'or. Les blocs sont minés à un rythme constant (144/jour), et associés à une prime (la richesse du minerai) qui est divisée par deux (le minerai devient moins riche en fonction de la profondeur) tous les 210,000 blocs (soit 4 ans). La prime initiale en 2009 était de 50 BTC. Le bitcoin est divisé en 10^8 satoshis. Pour ces raisons le nombre de bitcoin émis en 33 ans est limité à 21 millions (210,000 x 50 x 2), dont la moitié (10,5 millions) ont été crées au terme des quatre premières années de vie du système (soit fin 2012).

Ethereum et smart contract

Le concept de la blockchain Ethereum a été introduit[24] par Vitalik Buterin in 2013. Par la suite le projet s'est développé dans une société suisse (*Ethereum Switzerland GmbH*), et une fondation à but non lucratif (*Stiftung Ethereum*). Le système est opérationnel depuis le 30 juillet 2015. Cette blockchain est associé à une crypto monnaie, l'Ether (1 Ether=10$ en 2017), qui est divisé en 10^{18} Weis. Dans le minage du bitcoin les machines rigs interprètent des scripts de signature du payeur et d'identification de la clé publique du payé. Le langage utilisé n'est pas Turing complet, car il ne possède pas de boucles. Ethereum[25] introduit un langage Turing complet, basé sur des instructions identifiées par un octet. Ce langage permet d'intégrer à des transactions, l'exécution (appels) de programmes nommés Smart Contracts.

---

[24] Vitalik Buterin, "A Next-Generation Smart Contract and Decentralized Application Platform", 2013; https://github.com/ethereum/wiki/wiki/White-Paper

[25] "Ethereum: A Secure Decentralised Generalised Transaction Ledger", août 2015 - Yellow Paper

De manière similaire au bitcoin les utilisateurs sont identifiés par une adresse dérivée d'une clé publique (ECCC Sepc256k1), et possèdent une clé privée utilisée pour la signature des transactions. Les blocs sont minés toutes les 14,0 secondes, gratifiés d'une prime de 5 Ethers, à laquelle s'ajoute la notion d'oncle (une prime de (7-rang) x 5/8 d'Ether, avec rang >=0). La production de monnaie est donc constante (6000 blocs/ jour, 30,000 Ethers/jour). Contrairement au bitcoin, Ethereum stocke les transactions dans la blockchain. Ethereum utilise un *Proof of Work* (PoW) basé sur la résolution d'un problème difficile (*Ethash*, dérivé de le la procédure sha3-512), mais propose comme alternative à la consommation d'énergie, un algorithme de consensus nommé *Proof of Stake*. Ethereum revendique la définition d'un algorithme PoW difficile à intégrer dans des ASICs, afin de maintenir un hash rate modeste (8000 Gh/s en 2017). Une machine XFX R9 390, d'un prix de 450€, consomme 400W et produit 29 Mh/s; soit une consommation électrique journalière de 2,700,000 KW/h pour un cout de 400,000 €.

Les contrats sont stockés par des adresses dédiées. Chaque instruction de la machine virtuelle est facturée 1 Gas. La côte proposée du Gas en Ether (*Gas Price*) est insérée dans la transaction, ainsi que le cout maximum de l'exécution du contrat (*Start Gas*)

Sécurité des Applications Distribuées, Emergence de la Cyber Sécurité

La sécurité globale s'appuie le triptyque:

1) La sécurité des applications clientes et serveurs,

2) La sécurité des plateformes informatiques qui exécutent ces applications, et qui comportent deux composants:

- Le système d'exploitation

- Le hardware

3) la sécurité des protocoles de communication et du réseau qui transportent les messages d'applications

Sécurité des applications

Une application est une suite d'instructions (consommation de temps CPU) associée à contexte mémoire, qui s'appuie sur des ressources fournies par le système d'exploitation. Au sens TCP/IP les applications réalisent des services clients/serveurs, par exemple WEB (réseaux sociaux...) ou courrier électronique. Elles définissent généralement des modèles d'identités

permettant d'établir une politique de contrôle d'accès et des canaux sécurisés. Elles accèdent aux ressources du système d'exploitation partagés par différentes entités logicielles telles que fichiers, bases de données, bibliothèques réseaux ou cryptographiques. Le WEB utilise typiquement un mécanisme d'authentification multimodale, certificat et protocole TLS côté serveur et HTTP et mot de passe côté client.

Même si l'on peut concevoir des applications gérant intégralement leur sécurité, le maintien d'un secret incassable (non extractible) dans un logiciel reste en 2017 sans solution éprouvée; les tentatives infructueuse de développement des techniques WBC (*White Box Cryptography*) en sont une illustration.

Sécurité des protocoles de communication et des réseaux

Les piles réseau supportent des protocoles de sécurité par exemple IPSEC, SSH pour des couches TCP/UDP/IP ou IEEE 802.11i, IEEE 802.1x, RADIUS, PPTP pour les infrastructures LAN ou WLAN. Des procédures cryptographiques associées fréquemment à des preuves de protocole réalisent la confidentialité et l'intégrité des informations transportées. Les couches réseau s'appuient classiquement sur des ressources gérées par le système d'exploitation, telles que des bibliothèques cryptographiques ou des pilotes de circuit de communication. Elles offrent une surface d'attaque au niveau MAC ou TCP/IP, mais dont les défauts sont corrigés au fil du temps en particulier grâce à une standardisation d'usage des piles logicielles réseau. Des systèmes d'exploitation tels que Windows ou Android supportent une politique d'accès à ces ressources selon une granularité plus ou moins fine, sous forme de firewall d' applications ou d'autorisations. La gestion des identités et le confinement des secrets cryptographiques associés, sont les piliers sécuritaires de l'établissement d'échanges sécurisés. Les architectures réseaux réalisent également des concepts d'isolation physique et de cloisonnement, basés sur des nœuds intégrant des fonctions de firewall, de routage ou d'analyse de trafic (Deep Packet Inspection, DPI)

Sécurité du système d'exploitation

Le système d'exploitation organise l'isolation des processus et des données, par exemple à l'aide de mécanismes *sandbox*, et grâce à la définition d'une politique de contrôle d'accès aux fichiers DAC (*Discretionary Access Control*), MAC (*Mandatory Access Control*) mis en œuvre par SELinux (*Security Enhanced Linux*), ou RBAC (*Role Base Access Control*) utilisé par Windows et Unix. L'intégrité du système d'exploitation, c'est à dire la résistance à une modification non souhaitée de son code, typiquement

chargé en mémoire au démarrage du système (boot) est un problème critique. Il est partiellement solutionné avec des technologies dites de *Secure Boot*, basées sur des puces électroniques TPM (*Trusted Platform Module*). La réalisation des politiques d'accès s'appuie sur un système d'identité basé sur des tuples login/password ou certificat/clé-privé. La protection des données stockées par des mémoires secondaires (disques, flash...) implique la définition de système de gestion de fichiers chiffré dont la clé maitre est protégée par un mot de passe (typiquement celui de l'administrateur de la machine). Le stockage sécurisé d'un mot de passe est vulnérable à des attaques par force brute. De manière générale le contenu des mémoires primaires (SRAM, DRAM...) est en clair et peut stocker des informations sensibles collectées par divers processus.

Sécurité Hardware

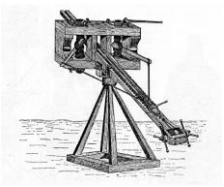
Le hardware classique s'appuie sur des portes logiques CMOS, des bus d'interconnexion, et implicitement sur des lois physiques conformes aux équations de Maxwell, qui sont sensibles à des attaques par canaux cachés (SPA, DPA...). Dans ces conditions le stockage de secrets (*Key Caching*) est très difficile. Des circuits électroniques équipés de contre mesures physiques et logiques (TPM, carte à puce, Secure Element, Secure MCU...) sont de plus en plus présents dans les systèmes informatiques, par exemple pour la restitution d'une clé maitre de chiffrement de fichiers liée à un mot de passe. Le confinement dans un espace sure (*tamper resistant*) évite/limite les attaques par force brute (grâce à l'absence de vecteurs d'attaque, par exemple empreinte du mot de passe), il impose également un accès physique à la puce de sécurité, et peut impliquer des temps d'accès "importants" qui rendent inefficaces les attaques force brute. De surcroit le blocage fonctionnel de la puce au terme d'un nombre fini de présentation du mot de passe (PIN code) peut entrainer la non fonctionnalité permanente de la plate forme informatique.

Quelques paradigmes de sécurité

Classification des types d'attaques

On peut classer[26] les attaquants d'un système en trois catégories:

[26] DG Abraham, GM Dolan, GP Double, JV Stevens, "Transaction Security System", in *IBM Systems Journal* v 30 no 2 (1991) pp 206 229



- **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 (hackers), mais avec des moyens limités. L'attaquant réalise un investissement modeste, mais espère un gain financier ou de notoriété.

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

Heuristiques de défense: placebo, vaccin, défense immunitaire

- **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.

Facteurs de Vulnérabilité: Complexité, Extensibilité, Connectivité

Trois facteurs[27] (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.

---

[27] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in Embedded Systems: Design Challenges", 2004.

Une Méthodologie de Cyber Attaque: Empreinte, Collecte, Inventaire

Dans les différentes éditions du best seller[28] "*Hacking Exposed*" les auteurs distinguent les trois phases suivantes pour la préparation d'une cyber attaque:

- *Le footprinting*, c'est à dire la collecte d'informations publiques sur les composants du système, via internet ou des services tels que WHOIS, DNS ou TRACEROUTE. Dans cette première phase le but est d'identifier les serveurs, leurs adresses IP, la structure du réseau, mais aussi les sites géographiques.

- *Le scanning* réalisant la collecte d'informations relatives aux systèmes d'exploitation, et aux ports TCP et UDP ouverts.

- *L'inventaire (enumeration)* consiste à obtenir des informations précises sur les services disponibles et leur version, collectées à l'aide de sessions actives.

Systèmes Embarqués

Les attaques sur les environnements SCADA, CAN, ou de manière plus générale sur les objets connectés, mettent en évidence les trois principes de sécurité suivants :

- **Communications Sécurisées** (intégrité, chiffrement...), avec mutuelle authentification forte. Cela implique la disponibilité de clés symétriques ou asymétriques, et la définition d'un système d'identité (basé sur des numéros de série ou des certificats par exemple)

- **Stockage Sécurisé**, par exemple pour les secrets nécessaires aux communications, mais plus généralement pour des informations sensibles.

- **Intégrité des Nœuds**, mécanismes d'isolation logicielle (par exemple multi processeurs pour la résistance aux malwares, SandBox pour l'intégrité du système d'exploitation), mise à jour/chargement de logiciels sécurisée (signature des logiciels), prévention détection des intrusions et comportements suspects, prévention des attaques par rebond.

[28] "Hacking Exposed", Stuart McClure, Joel Scambray, George Kurtz, Mac graw Hill

Principes de sécurité.

Classiquement la sécurité s'appuie sur cinq concepts de base, l'identification, l'authentification, la confidentialité, l'intégrité, et la non répudiation. La disponibilité est également importante.

Identification

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…

Authentification

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.

Confidentialité

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, 16 octets pour l'AES).

Intégrité

L'intégrité des données (MAC, Message AuthentiCation). Le chiffrement évite les écoutes indiscrètes, 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 AuthentiCation*. Le chiffrement et l'intégrité peuvent être combiné dans un seul algorithme AEAD (*Authenticated Encryption with Associated Data*), par exemple AES-CCM, AES-GCM.

Non-répudiation

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 privée de son auteur (par exemple RSA(Empreinte(Message))).

Disponibilité

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

Au sujet de la confiance

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»).

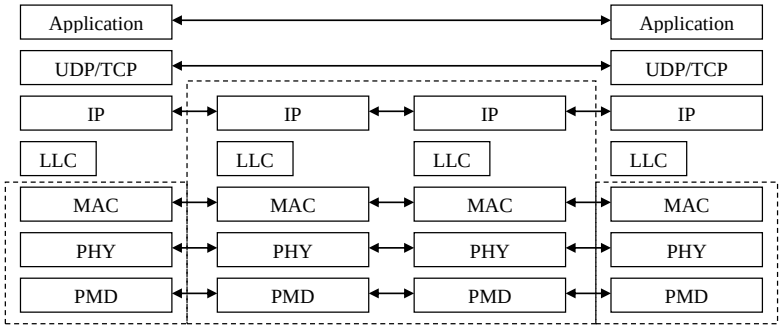
(Client R CA) ET (Serveur R CA) => (Client R Serveur)

Sécurité des Réseaux

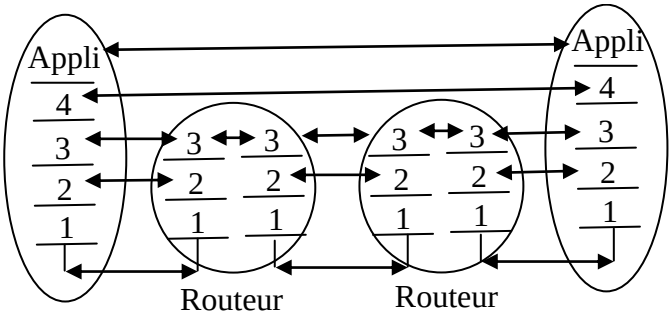
Un réseau assure le transport des messages échangés entre deux applications distantes. Dans le modèle OSI les services déployés par le réseau sont classés en sept couches, physique, données, réseau, transport, session, présentation et application. Le modèle classique des réseaux TCP/IP

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, dédiés aux couches OSI physique, MAC, TCP/IP, application, ou intégré entre transport et application.

Sécurité au niveau physique

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.

Sécurité au niveau MAC

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…).

Sécurité au niveau réseau/transport

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.

Couche de Sécurité entre transport et application

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/TLS 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

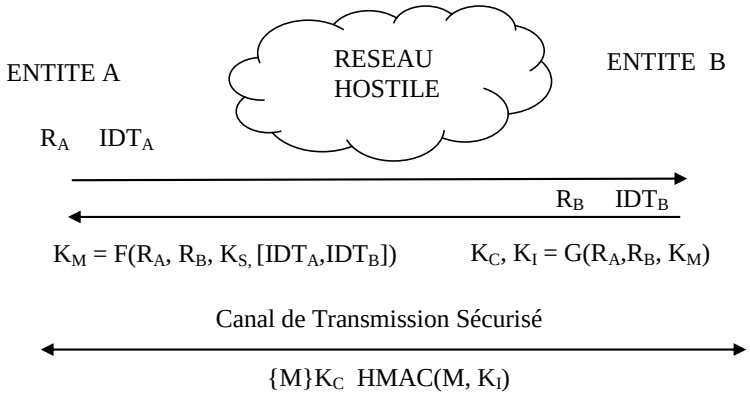
navigateur et un serveur bancaire, le client authentifie son service bancaire. Une fois le tunnel sécurisé établit le client s'authentifie à l'aide d'un login et d'un mot de passe. Il obtient alors une identité temporaire associée à un simple cookie.

Sécurité au niveau application

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.

Eléments de Sécurité

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 à long terme) 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, FIDO…

- **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

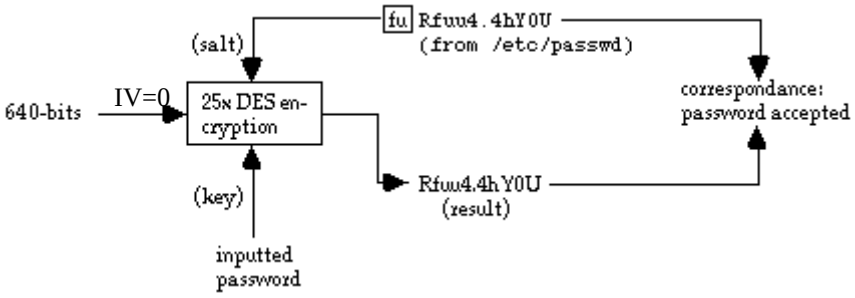
Nous classons les méthodes d'authentification en trois catégories, symétriques (secrets partagés), asymétriques (basées sur RSA ou ECC en règle générale) et tunnels.

Mécanismes symétriques: mot de passe, pre-shared key, provisionning

Nous divisons les méthodes symétriques en trois classes

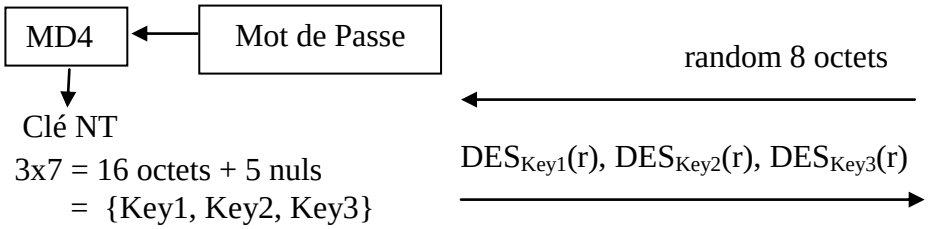
- **Les mots de passe**. Un mot de passe est une suite de caractères qui peut être facilement mémorisée par un être humain. Il ne s'agit pas d'un nombre aléatoire mais au contraire d'une concaténation de mots, de chiffres et de signes. De tels secrets peuvent être devinés à l'aide d'une attaque par dictionnaire, 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 ficher /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 x 7bits). 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.



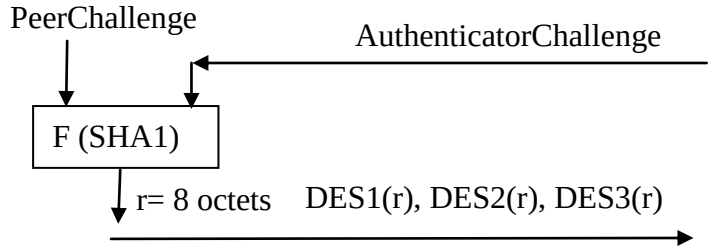
Dans l'univers Microsoft la sécurité d'un ordinateur personnel est fortement corrélée au mot de passe de son utilisateur. Ce dernier n'est jamais stocké en clair dans la mémoire de la machine. A partir d'un mot de passe

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.



Sous Windows le SAM (*Security Account Manager*) est la base de donnée qui contient les informations sur les comptes utilisateurs (login, mot de passe). L'image du SAM est stockée dans le fichier C:\Windows\System32\config et également dans la base de registre HKEY_LOCAL_MACHINE\SAM. Cependant ces informations ne sont pas accessibles au *runtime*. On obtient un copie du SAM en démarrant la machine sous un autre système d'exploitation (typiquement linux), mais aussi à l'aide d'utilitaires de hacking tels que fgdump.exe (exécuté en mode

administrateur). Selon[29] dans la méthode *LM Hash* le mot de passe d'un compte utilisateur est complété à 14 octets à l'aide d'octets nuls, puis transformé en une deux clés DES distinctes réalisant le chiffrement de la valeur ASCII KGS!@#$% (soit 2x8 octets). La procédure *NTLM Hash* stocke l'image MD4 (16 octets) du mot de passe.

Des programmes tels que *CAIN* ou *Windows Password Recovery* réalisent des attaques par force brute, dictionnaire ou rainbow tables.

- **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 sûres (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 indiscrètes.

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,

[29] https://technet.microsoft.com/en-us/library/cc875839.aspx, "Selecting Secure Passwords"

en raison des nombreux logiciels disponibles sur le WEB qui cassent les protocoles à base de mot de passe.

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 démuni 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 $2^{32}/2^{14}*2^{10} = 2^{28}$.

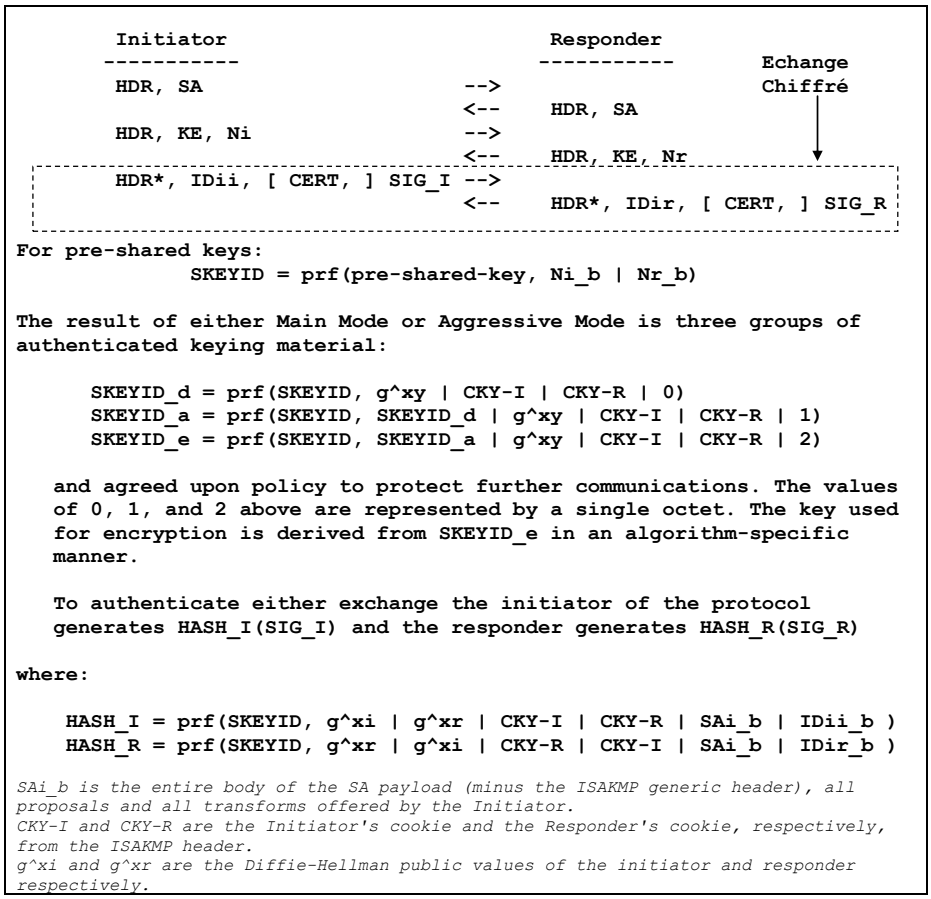
De nombreux protocoles (PPP, POP, FTP,…) mettent en œuvre des mots de passe transmis en clair sur le réseau. Certaines messageries ne supportent pas un transport sécurisé par SSL/TLS.

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

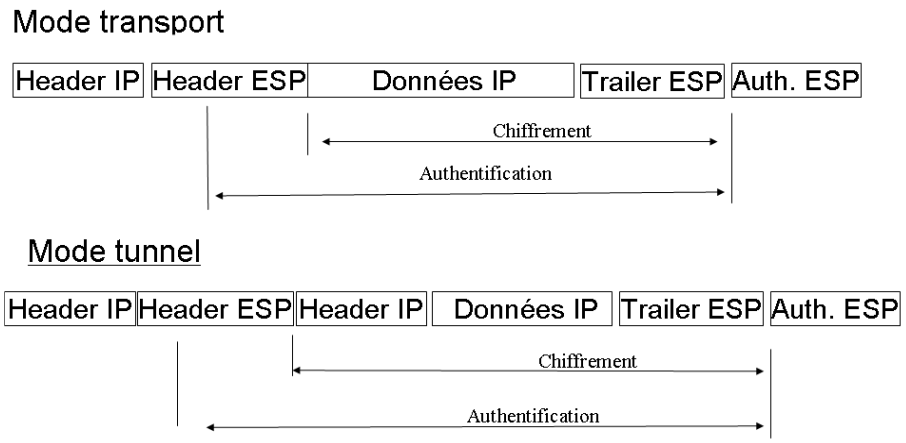
Les solutions sécurisées IP classiques: VPN, TLS, Kerberos, SSH

IPSEC

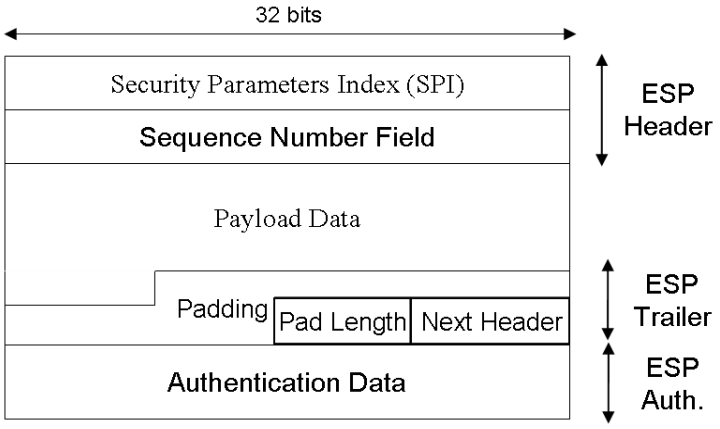
IPSEC assure l'intégrité, l'authentification et la confidentialité des charges transportées par le protocole IP. Les ressources cryptographiques sont déduite d'une **association de sécurité** (SA) basée sur un secret partagé entre les deux entités de la session sécurisée IP. La difficulté de déploiement du protocole IPSEC est liée au mécanisme d'établissement d'un SA, à l'aide du protocole IKE (*Internet Key exchange*, 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, *preshared key main mode*.



Mode Transport et Tunnel dans le protocole ESP, IPSEC

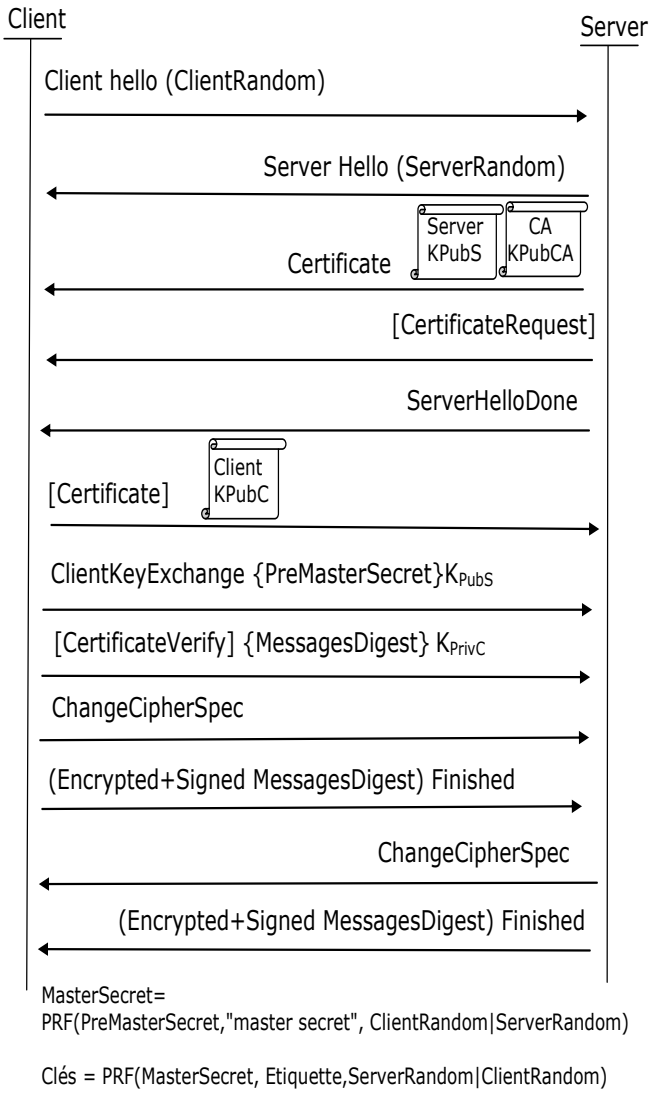


En tête du protocole ESP.

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 la vérification des certificats serveurs.

Kerberos

Kerberos est un protocole développé par le MIT (*Massachusetts Institute of Technology*) 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…).

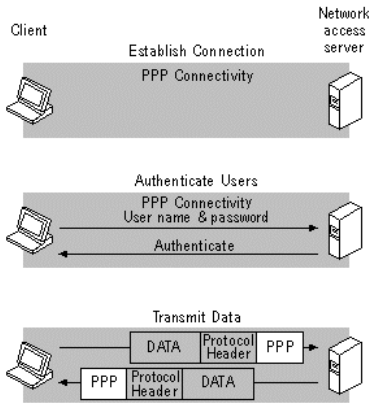
Le serveur d'authentification réalise l'authentification d'un utilisateur à l'aide de son mot de passe; il délivre un ticket d'authentification, *Ticket Granting Ticket*, ou TGT, identifié par un principal. Le *Ticket Granting Server* (TGS) génère des tickets de service pour un utilisateur authentifié,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

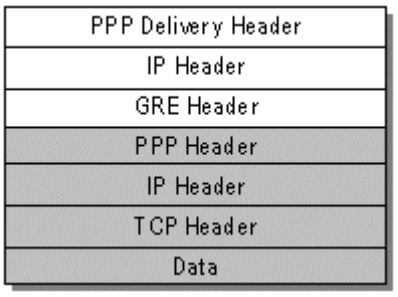
La clé de session (*Session Key*) est dans le cas d'un utilisateur du KDC une clé cryptographique (DES, 3xDES, AES) déduite de son mot de passe; et dans le cas d'un service elle est générée par le KDC.

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.

PPTP-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 IP 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.





SSH

La première version de SSH (SSH-1) fut conçue par Tatu Ylönen, à Espoo, en Finlande en 1995. La version suivante a été nommée SSH-2. Le groupe de recherche de l'IETF *secsh* a défini en janvier 2006 le standard Internet SSH-2, par un jeu de RFCs

RFC 4251, Secure Shell (SSH) Protocol Architecture

RFC 4253, The Secure Shell (SSH) Transport Layer Protocol

RFC 4252, The Secure Shell (SSH) Authentication Protocol

RFC 4254, The Secure Shell (SSH) Connection Protocol

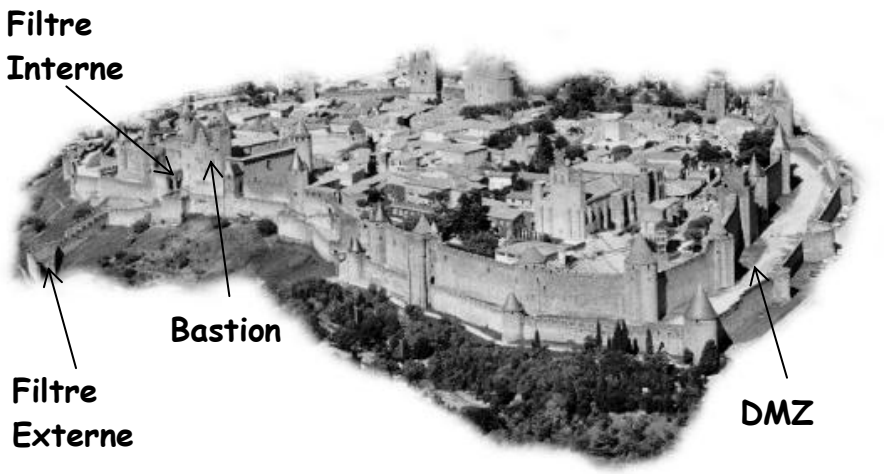
Le document "Protocol Architecture" (RFC 4251) décrit les différents éléments de SSH:

- Les messages SSH sont transportés par des paquets TLP (Transport *Layer Protocol*). TLP réalise l'authentification du serveur, la confidentialité et l'intégrité des messages SSH. Le calcul des clés est basée sur un échange Diffie-Hellman, dont les paramètres sont certifiés par une signature DSA généré par le serveur.

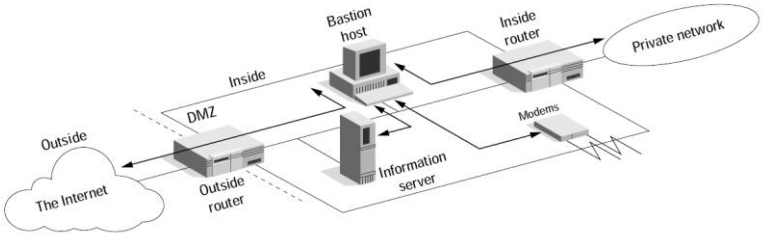
- Le bloc *User Authentication Protocol*, est en charge de l'authentification du client, généralement grâce à un couple login/password, mais il est possible d'utiliser un certificat et la clé privé associée. Les messages sont acheminés via le canal sécurisé mis en place par TLP.

- Le *Connection Protocol*, multiplexe plusieurs canaux logiques (Shell X11...) dans le tunnel sécurisé. Les messages sont transportés par le *User Authentication Protocol*.

Limitation des protections offertes par les pare-feu.



Un pare-feu[30] 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

[30] 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.

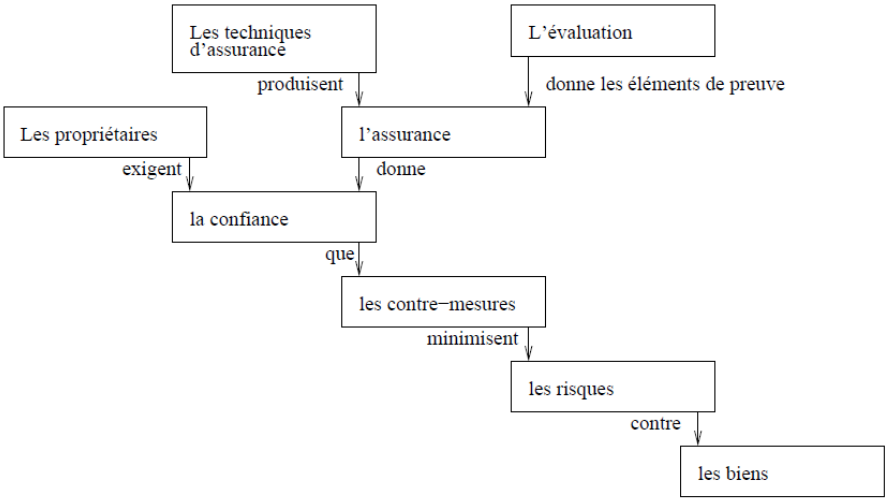
Au sujet des normes Critères Communs (CC)

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.

Heuristiques d'Attaque

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.

James P. Anderson[31] 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.

[31] James P. Anderson, Computer Security Threat Monitoring and Surveillance, James P. Anderson Co, Fort Washington, PA (1980).

Les virus sont répartis[32] selon quatre catégories principales, les virus systèmes, les virus programmes, les virus interprétés, et les vers.

Les virus systèmes sont dominants au début des années 90. il s'installent dans les secteur de BOOT des disquettes ou dans les MBR (*Master Boot Record*) des disques durs.

Les virus programmes sont stockés dans des fichiers exécutables et se propagent dans de multiples fichiers.

Les virus interprétés se propagent dans des macro (par exemple utilisés dans Microsoft Office) ou des scripts (par exemple JavaScript ou VBSript).

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

Les anti virus mettent en œuvre des méthodes de détections basées sur :

- la recherche de signature (un ensemble d'instructions, c'est une méthode peu efficace avec des virus polymorphes)

- la recherche générique (recherche de sous ensembles d'instructions)

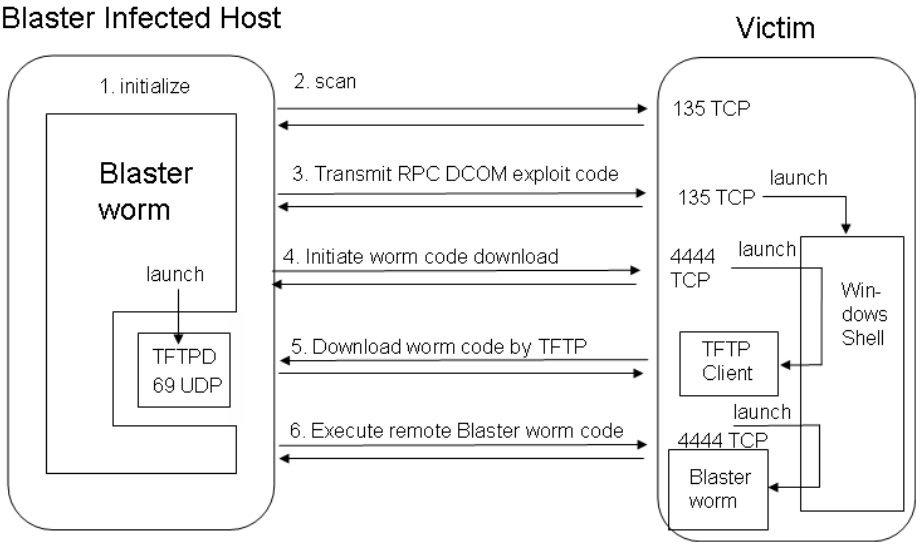
- la recherche heuristiques (ensemble d'instructions singulières et suspectes)

- le monitoring d'un programme (détection d'opérations non autorisées ou suspectes)

Le premier vers (IBM Christmas Tree) a été écrit par un étudiant Ouest Allemand en 1987 et diffusé sur la messagerie VNET. Un courrier comportait un programme attaché *christma.exec* qui affichait des vœux et rediffusait le message à l'ensemble du carnet d'adresse du poste client.

[32] Vers et Virus, François Paget, DUNOD 2005

Exemple le ver Blaster (2003)



Principe du vers «*Blaster*» (2003)

Au sujet des botnets

Le terme botnet[33] 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).

[33] 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).

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 chaines (char[]) de caractères dont la marque de fin est un octet nul. Les fonctions qui manipulent ces chaines (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.

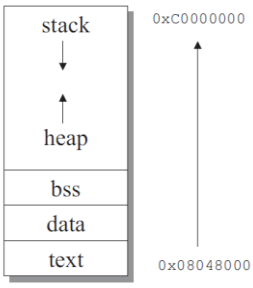
Les effets espérés par les *buffer overflow* dans la pile d'exécution sont répartis en trois classes principales

- 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.

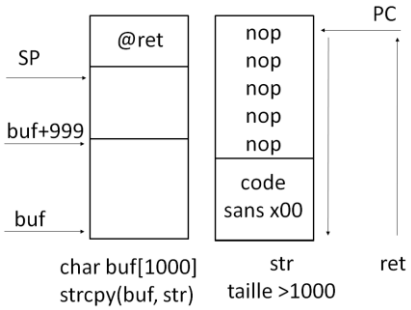


Par exemple le format ELF (*Executable Linking Format*), utilisé par de nombreux systèmes UNIX, (et Windows) est un exemple de l'architecture dite de *Van Neumann* dans laquelle le code et les données sont logés dans une mémoire commune. Il existe un autre concept, l'architecture d'*Harvard*, dans lequel le code et les données sont stockés dans des mémoires distinctes; cette dernière approche est usuelle dans les environnements de type microcontrôleurs.

Le format ELF 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*.



L'ouvrage[34] "Techniques de Hacking", de Jon Erickson, détaille les techniques d'exploitation de *buffer overflow* et l'injection de code en particulier de *Shell Code*. La zone de données écrite à partir de l'adresse de base du buffer comporte un code malicieux (sans octet nul) et une série d'instructions NOP (no operation) qui écrasent l'adresse de retour du compteur programme (PC) mémorisé dans la pile. Ce code tente une escalade de privilège (via la procédure *setresuid()*), afin d'exécuter un shell (tel que /bin/sh) en mode superviseur (root).

34 "Techniques de Hacking", Jon Erickson, Pearson, 2009-2012

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 CSS

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

Cross Site Request Forgery CSRF

Une attaque CSRF 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.

Autour de la Sécurité des Systèmes d'Exploitation

Stratégie de défense du système Windows (2005)

Voici un bref résumé de la stratégie de sécurité d'un grand éditeur de logiciels (Microsoft, RSA Security 2005).

- 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.

Principes de sécurité du système Android

Android est un système d'exploitation crée 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 librairies natives, l'environnement de la Dalvik Virtual Machine (DVM), le framework Java (JINI), 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 *quatre* piliers :

- Les SANDBOXs Unix, associés aux processus

- Les permissions.

- La signature des applications

- le chiffrement des fichiers.

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.

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***) ou de WBC (*White Box Cryptography*) peuvent rendre ces attaques plus difficiles.

Le groupe *Trusted Computing Group* (TCG) a défini une architecture sécurisée pour les ordinateurs personnels basés 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.

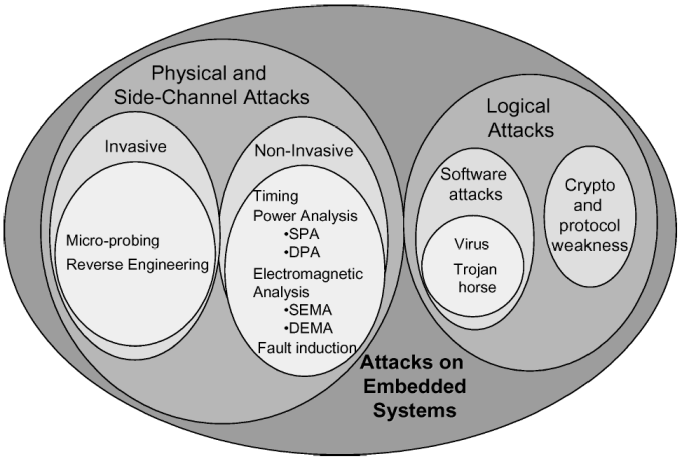
L'intégration du TPM dans le système d'exploitation *Windows* s'applique aux trois points suivants:

- Le *Secure Boot*. Le premier programme amorce est stocké dans le TPM. Le *boot* est une succession de programme Pi tels que Po est contenu dans le TPM, chaque Pi est associé à une empreinte Hi enregistrée dans le TPM, le programme Pi-1 charge Pi et vérifie son empreinte Hi.

- Le chiffrement du contenu du disque dur (*bitlocker™*) 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*).

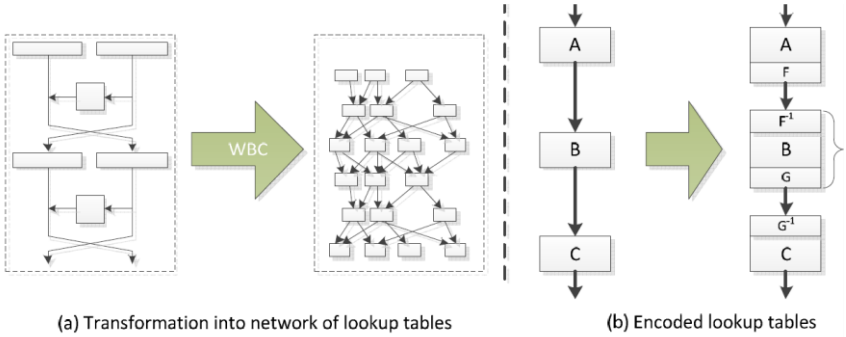
Exemple de classification des attaques pour les systèmes embarqués.

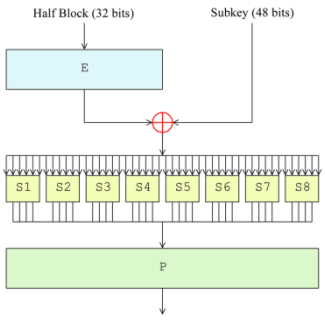


Examples of attack threats faced by embedded systems.

S. Ravi et al, 2004

White-Box Cryptography (WBC)





La technique de White Box a été proposée en 2002[35]. Son objectif[36] est de réaliser l'obfuscation logicielle d'une clé associée à un algorithme cryptographique par exemple DES ou AES. Le code implémentant l'algorithme peut être cloné, mais l'extraction de la clé est présumée impossible. L'algorithme de chiffrement est transformé en un réseau de tables (*lookup table*) liées à la clé (S-Box modifiées).

Une table S établit une relation entre un index (6 bits pour le DES) et une sortie (4 bits pour le DES), $index_6 = K_6$ exor E_6 (K_6 est une sous clé DES de 6 bits), $Y_4 = S(index)$. Soit un $index_6' = index_6$ exor K_6, on obtient une table $Y_4 = S(index_6')$. En appliquant la relation $index_6'$ exor E_6 on retrouve $index_6$, et donc $Y_4 = S(index_6'$ exor $E_6)$. Le code est par la suite embrouillé selon des techniques d'obfuscation.

Les implémentations DES et AES en white box ont été cassées[37] par des attaques dites DCA (*Differential Computation Analysis*).

Lors de la conférence BlackHat une série d'attaques[38] (injection de fautes et analyse *Differential Fault Analysis* - DFA) a été démontrée sur des implémentations commerciales.

---

35 Stanley Chow & All. "A white-box DES implementation for DRM applications", in Proceedings of the ACM Workshop on Security and Privacy in Digital Rights Management (DRM 2002), volume 2696 of Lecture Notes in Computer Science, pages 1–15. Springer, 2002.

36 Brecht Wyseur, Nagra Kudelski, "White-Box Cryptography: Hiding Keys In Software", MISC 2012.

37 Joppe W. Bos, et Al, "Differential Computation Analysis: Hiding your White-Box Designs is Not Enough"; 2016.

38 Eloi Sanfelix, Cristofaro Mune, Job de Haas, "Unboxing the White Box", BlackHat 2016

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,

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

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

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

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

En constate que, dans cette implémentation de l'algorithme RSA (dite *exponentiation*), chaque bit (ei) de la clé implique un temps calcul différent selon que sa valeur soit 0 (multiplication triviale par 1) ou 1 (multiplication par $M^{2\wedge i}$). En fonction des différences de temps calcul observées on déduit la valeur de ei (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 (k étant compris entre 0 et p-1) prend comme argument d'entrée une valeur d'entré $M_{k,i}$ et calcule la valeur de sortie $M_{k+1,i}$.

$M_{0,i}$ est un message en clair 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^{nk} - 1).

Par exemple dans le cas de l'algorithme DES chaque étage utilise une (sous) clé de 6 bits (nk = 6). L'algorithme travaille avec une clé globale de 56 bits, mais cette dernière est appliquée sur différents blocs de calcul 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^{nk}-1).

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

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^{nk} produits $P_{k,j,i}(t)$

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

On obtient 2^{nk} 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.

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

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.

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

Faites moi confiance ? 20 ans de bugs et heuristiques

Voici une liste non exhaustive de failles de sécurité.

- 1995, Peter Shor[39] invente un algorithme de factorisation d'un nombre N par un calculateur 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 (3x5) en utilisant un calculateur quantique de 7 qbits.

- En juillet 1995, Damien Doligez, doctorant à l'INRIA a cassé un clé RC4 de 40 bits en une semaine, à l'aide d'une centaine de machines générant chacune de l'ordre de 10000 clés/seconde.

- 1996, Lov K. Grover[40] 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 la société Google associée à la NASA a fait l'acquisition d'un ordinateur quantique (D-WAVE) de 8x8 qbits (quantum bits).

- 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[41].

- 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[42] cassent le WEP en environ 2^{22} (4 millions) d'essais.

---

[39] Peter W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", 1995.

[40] Lov K. Grover "A fast quantum mechanical algorithm for database search", 1996

[41] Marc Briceno, Ian Goldberg, and David Wagner

[42] 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.

- 2002, trois scientifiques indiens[43] 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'ou 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).

- En 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*.

[43] Manindra Agrawal, Neeraj Kayal and Nitin Saxena "PRIMES is in P", 2002

- En 2008, Jeroen van Beek, modifie avec succès le RFID des passeports électroniques Hollandais. L'attaque repose sur deux concepts, réalisation 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[44], 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 connait 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[45], 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*".

---

44 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

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

- Fin 2013, un malware (BlackPOS) logé dans des terminaux de paiement (POS) des magasins TARGET réalise le détournement de 40 millions de carte de crédits.

- En 2013, un document publié par Edward Snowden révèle que les échanges de données entre les Data Center de la société Google ne sont pas chiffrés.

- en 2014 un groupe[46] de chercheurs français publie un algorithme quasi polynomial réalisant le calcul d'un logarithme discret dans les corps finis, de complexité

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

Pour des corps fini tels que

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

ou pour des corps de faibles caractéristiques (typiquement 2) $\mathbb{F}_{2^n}$

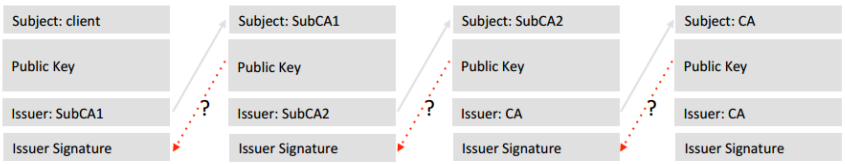
soit une complexité de $2^{O((\log n)^2)}$

- En 2014 Jeff Forristal directeur technique de la société Bluebox a publié lors de la conférence[47] 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 chaine n'est pas vérifiée.

---

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

47 https://www.blackhat.com/docs/us-14/materials/us-14-Forristal-Android-FakeID-Vulnerability-Walkthrough.pdf



- 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,
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;
}
```

- En 2014 Karsten Nohl (& All) a démontré[48] lors de la conférence BlackHat 2014 que les mises à jour de firmware des jetons USB ne sont pas sécurisées. Selon la norme USB un jeton peut comporter plusieurs fonctions par exemple stockage mais également clavier (*keys logging*) ou interface réseau (écoute du trafic)

- En 2015 l'article[49] démontre une attaque nommée Logjam qui calcule un logarithme discret de p=512 bits dans Z/pZ en une minute, après une phase de pré calcul sur le groupe d'environ une semaine. La complexité de l'attaque est résumé par la relation ci dessous avec N=2^p et k compris entre 1 et 2.

$$\exp\left((\hat{k} + o\check{(}1))(\log N)^{1/3}(\log\log N)^{2/3}\right)$$

soit exp (1 * 7,8 * 9) = $2^{1,45*7,8*9} = 2^{100}$

- En septembre 2016 le malware *Mirai* se loge dans 145.607 caméras de la société chinoise *XiongMai Technologies*,, dans plus de 100 pays, et réalise des attaques DDOS. Il génère 1 terabit/s de trafic IP, et produit environ 35,000/50,000 requêtes HTTP par secondes. Il utilise une faille de sécurité triviale des caméras (une URL permettant d'injecter du code, http://<IP_address_of_device>/DRV.htm).

- En 2016 Tobias Boelter[50], chercheur à l'université californienne de Berkeley, met en évidence un trou de sécurité de l'application WhatsApp[51], basé sur une technique MIM (Man In the Middle) possible dans des réseau GSM.

---

48 "BadUSB On accessories that turn evil ; KarstenNohl; Sascha Krißler; Jakob Lell.

49 "Imperfect Forward Secrecy:How Diffie-Hellman Fails in Practice" David Adrian et Al.

50 https://tobi.rocks/2016/04/whats-app-retransmission-vulnerability/

51 WhatsApp Encryption Overview, Technical white paper November 17 2016

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')

- **1st pré-image**, étant donné X, trouver M tel que H(M) = X

- **2nd 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}.

En 2015[52] les premières collisions SHA1 (plus précisément sur la fonction de compression Compr(IV1,M1)= Compr(IV2,M2))sont publiées par Marc Stevens, Pierre Karpman, et Thomas Peyrin. L'algorithme a une complexité de 2^{58} et nécessite 10 jours de calcul pour un cluster de 64 processeurs (*General Purpose Unit* - GPU).

---

52 "Freestart collision for full SHA-1"
Marc Stevens, Pierre Karpman, and Thomas Peyrin, 2015

Message 1																
IV_1	50	6b	01	78	ff	6d	18	90	20	22	91	fd	3a	de	38	71	b2	c6	65	ea
M_1	9d	44	38	28	a5	ea	3d	f0	86	ea	a0	fa	77	83	a7	36				
	33	24	48	4d	af	70	2a	aa	a3	da	b6	79	d8	a6	9e	2d				
	54	38	20	ed	a7	ff	fb	52	d3	ff	49	3f	c3	ff	55	1e				
	fb	ff	d9	7f	55	fe	ee	f2	08	5a	f3	12	08	86	88	a9				
Compr(IV_1,M_1)	f0	20	48	6f	07	1b	f1	10	53	54	7a	86	f4	a7	15	3b	3c	95	0f	4b

Message 2																
IV_2	50	6b	01	78	ff	6d	18	91	a0	22	91	fd	3a	de	38	71	b2	c6	65	ea
M_2	3f	44	38	38	81	ea	3d	ec	a0	ea	a0	ee	51	83	a7	2c				
	33	24	48	5d	ab	70	2a	b6	6f	da	b6	6d	d4	a6	9e	2f				
	94	38	20	fd	13	ff	fb	4e	ef	ff	49	3b	7f	ff	55	04				
	db	ff	d9	6f	71	fe	ee	ee	e4	5a	f3	06	04	86	88	ab				
Compr(IV_2,M_2)	f0	20	48	6f	07	1b	f1	10	53	54	7a	86	f4	a7	15	3b	3c	95	0f	4b

Quelques TOP10 d'attaques

Le TOP9 des menaces visant le cloud computing en 2013

La virtualisation est une technique 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[53] "*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 ExtractPrivate 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*).

53 https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf

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.

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) est une communauté publique permettant à des organismes de développer, acheter et maintenir des applications fiables. Il publie[54] chaque année un rapport sur dix les risques de sécurité WEB les plus critiques:

[54] http://www.owasp.org/index.php/Top_10

A1- Faille d'injection. Par exemple injection SQL, OS et LDAP. Elle se produit lorsqu'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. Elles permettent aux attaquants de compromettre les mots de passe, clés, jetons de session, ou d'exploiter d'autres failles d'implémentation afin de s'approprier les identités d'autres utilisateurs.

A3 – Cross-Site Scripting (XSS). Les failles XSS se produisent lorsqu'une application accepte des données non fiables et les envoie à un browser web sans validation appropriée. XSS permet à des attaquants d'exécuter un 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 d'un objet interne, tel que fichier, dossier, enregistrement de base de données, ou 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 sécurité pertinente implique une configuration sécurisée définie et déployée pour les applications, contextes, serveurs d'application, serveurs web, serveurs de base de données et la plateformes. Tous ces paramètres doivent être définis, mis en œuvre et maintenus, car beaucoup ne sont pas activés 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 réaliser un vol d'identité, de la fraude à la carte de crédit ou autres délits. Les données sensibles nécessitent une protection supplémentaire, par exemple un chiffrement, ainsi que des précautions particulières lors des échanges avec le navigateur.

A7 – Manque de contrôle d'accès au niveau fonctionnel. Pratiquement toutes les applications web vérifient les droits d'accès avant de rendre une

fonction 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 fonction 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 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.

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.

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.phonefactor.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 connu; 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 Ci représente les éléments chiffrés obtenus à partir des blocs en clair Mi. Dans le cas du 3xDES la taille de ces blocs est de 8 octets, soit 64 bits d'entropie (2^{64}). L'attaquant connait un prochain IV_{l+1} par observation des informations préalablement chiffrées. Il injecte la donnée IV_{l+1} exor IV_j 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

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

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 Ck) 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).

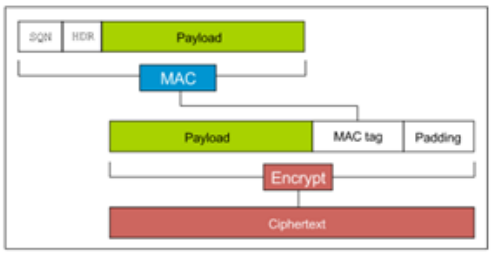


Figure 1: D(TLS) encryption process

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 le 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 octets + 256 x 0xFF (soit 256 octets de padding)

P1 = 287 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

M0 = SQN || HDR || P0 || T || pad

M1 = SQN || HDR || P1 || T || pad

L'attaquant observe

M = SQN || HDR || Pi || T || pad

Il forge

M' = SQN || HDR || Pi, ce message est traité par le protocole DTLS

Si la charge est égale à P0, 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 P1, 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 (P4) d'un bloc chiffré de 16 octets (C4), il a également observé la valeur C' utilisée lors d'un précédent chiffrement en mode CBC (dans la séquence C' || C4)

C4 = E (P4 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 (C1, C2, C'exor X, C4)

M= SQN || HDR || C1 || C2 || (C' exor X) || C4 (soit 64+13=77 octets)

Le protocole DTLS déchiffre la valeur (D= déchiffrement)

P4' = D(C4) exor (C' exor X) = (P4 exor C') exor C' exor X = P4 exor X

Si le dernier octet de P4' 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 (4x16= 64 = 43 + 20 + 1). Le calcul du HMAC s'applique donc à 56 octets

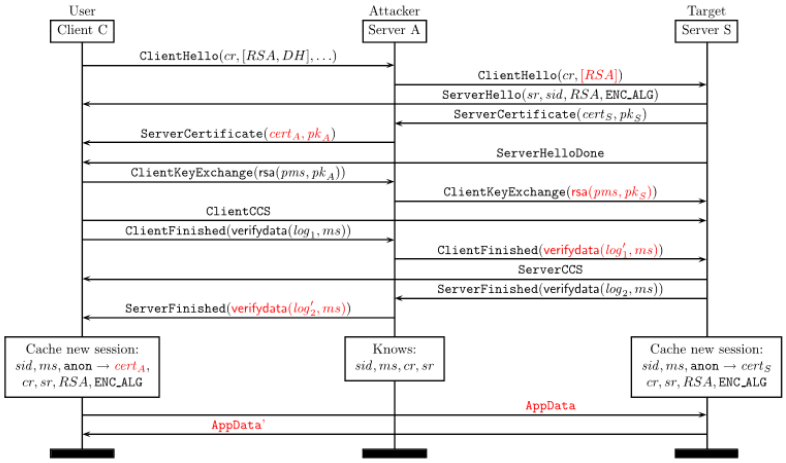
Si le dernier octet de P4' indique une longueur de padding dont la structure est correcte (par exemple 0x01 0x01, la suite la plus probable) le calcul du HMAC s'applique à 55 octets au plus. L'algorithme HMAC utilise deux blocs de 64 octets pour une longueur inférieure à 55 et au moins trois blocs de 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 (2014)

Cette attaque[55] a été publiée le 4 mars 2014 par une équipe de recherche de l'INRIA Rocquencourt. Elles un 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



55 https://secure-resumption.com/

HeartBleed (2014)

HeartBleed est un bug du logiciel OPENSSL détecter en 1014, qui permet de lire un bloc aléatoire de mémoire de 64ko.

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.

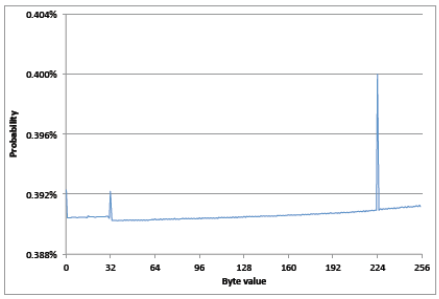
```
struct {
    HeartbeatMessageType type;
    uint16 payload_length;
    opaque payload[HeartbeatMessage.payload_length];
    opaque padding[padding_length];
} HeartbeatMessage;
```

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'a 64 Ko)

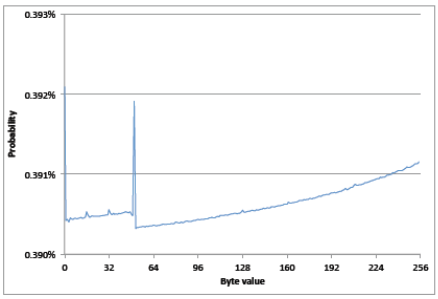
L'attaque RC4 du L'attaque Royal Holloway (2013)

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 >=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^{44} 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^{ième} octet a une probabilité de 0,4% d'avoir pour valeur 224



(c) Byte Z_{32}



(d) Byte Z_{50}

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 Ksi+1 au rang i+1 connaissant la valeur Ksi 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%.