Introduction
à la Cyber Sécurité 2023

TELECOM Paris

IP PARIS

Machine Enigma, *Imperial War Museum*, Londres

Pascal Urien - Télécom Paris

I-PREFACE

II-INTRODUCTION

III-PLATEFORME

VI-SYSTEMES

V-ELEMENTS DE SECURITE

VI-RESEAUX

VIII-QUELQUES ATTAQUES

Préface

Une tentative de définition de la Cybersécurité.

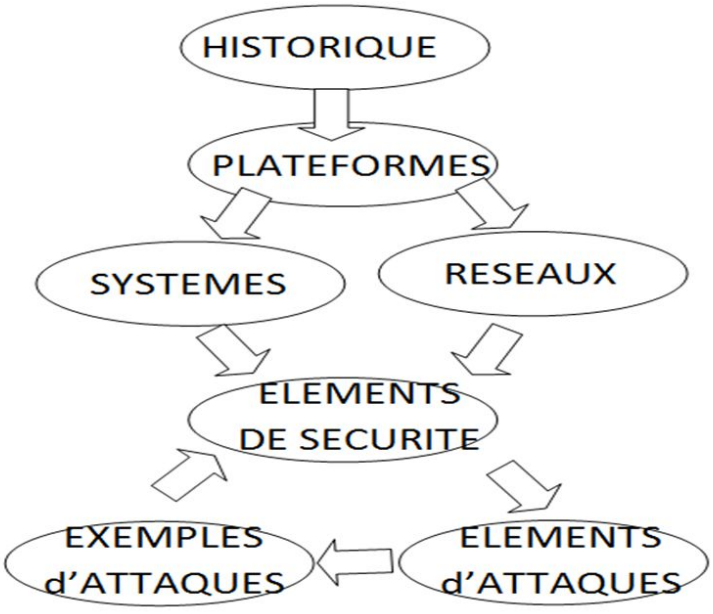
La cybersécurité vise à garantir le fonctionnement "normal" (*nonimal*) d'un système numérique.

Bien que ce fonctionnement normal ("*goodware*") soit difficile à définir formellement, l'idée est que le système reste conforme à un cahier des charges. Le but d'un attaquant est de modifier le comportement ("*behavior*") du système numérique et/ou d'accéder à des informations confidentielles.

Un système numérique est composé de processeurs, associés à des mémoires hébergeant des programmes et des données. Des canaux d'entrées sorties (I/O) permettent des interactions locales (clavier, écran…) ou distantes (réseaux…) avec des humains ou des machines.

Les applications numériques sont multiples, véhicules autonomes, supervision de processus industriels, commerce électronique, échange d'information multimédia, réseau domotique…

La surface d'attaque, et donc le périmètre de défense de la cybersécurité, est complexe à appréhender, les technologies (microélectronique, algorithmes, protocoles…) ayant évoluées sous la pression du marché, en l'absence de pré-requis de sécurité.



Introduction

Introduction

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

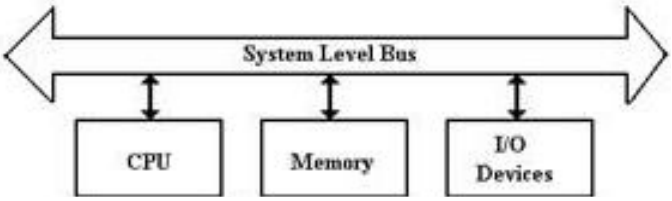
Ordinateurs et sécurité des applications

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 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 fondée en 1968 par les docteurs Gordon Moore, Robert Noyce et Andrew Grove, créa en 1971 le premier microprocesseur 4 bits (le 4004, qui comportait 2300 transistors). Un microprocesseur CISC (*complex instruction set computer*) est organisé autour d'une unité arithmétique et logique (ALU) qui exécute des micro-instructions (µI) stockées dans la mémoire du microprogramme (µP). Dans ce cas l'intégrité du µP est le *point zéro* de la confiance. Dans le modèle RISC (*reduced instruction set computing*) il n'y a pas de µP, les instructions sont exécutées directement par un décodeur d'instruction.

[1] Un des premiers ordinateurs, "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.



Le système PDP 11, 1970

Le PDP-11 (*Programmable Data Processor*) commercialisé par Digital Equipment Corporation (DEC) dans la période 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).

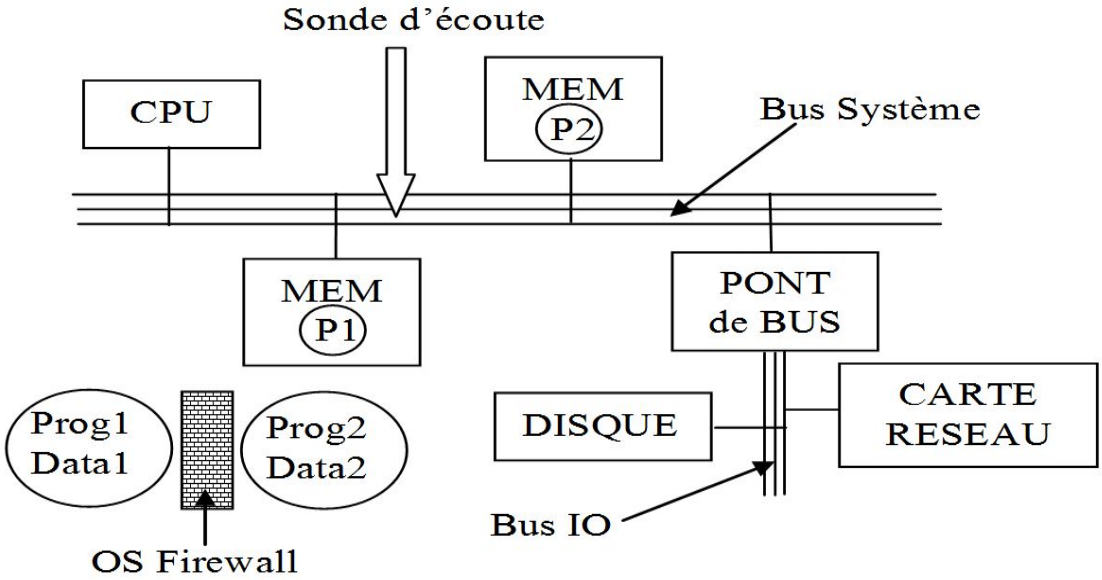
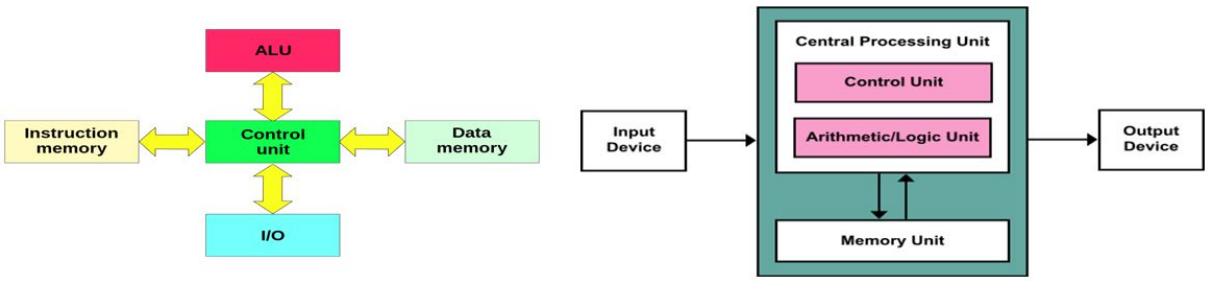


Illustration des attaques et défenses dans un système informatique classique. Le système d'exploitation réalise l'isolation logique des programmes. Le contenu des mémoires non chiffré, transite en clair sur le bus système.

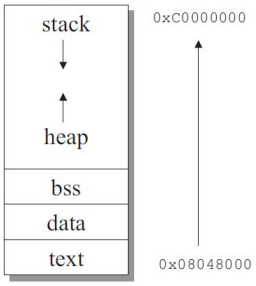
Le système d'exploitation CP/M (ancêtre du DOS), créé en 1974 par *Gary Kildall*, fondateur de *Digital Research* utilisait des lignes de commande selon la syntaxe *ProgramName FileIn FileOut*.



Architecture de Harward (à gauche) et Von Newmann (à droite)

Remarquons qu'il existe deux classes majeures d'architecture d'ordinateur: l'architecture de *Harvard* (systèmes embarqués…) dans laquelle les mémoires de données (RAM) et de code (FLASH) sont disjointes, et l'architecture de *Von Neumann* (Unix, Windows…) où le code et les données sont stockés dans une mémoire (RAM)

commune. L'injection de code est plus difficile dans le cas d'une architecture de Harvard, puisqu'il est nécessaire de stocker le code malveillant dans la mémoire non volatile (FLASH) des instructions. Par exemple une attaque "*buffer overflow*" injecte des instructions dans la mémoire RAM, l'exécution de code en RAM est impossible dans le modèle de Harvard.

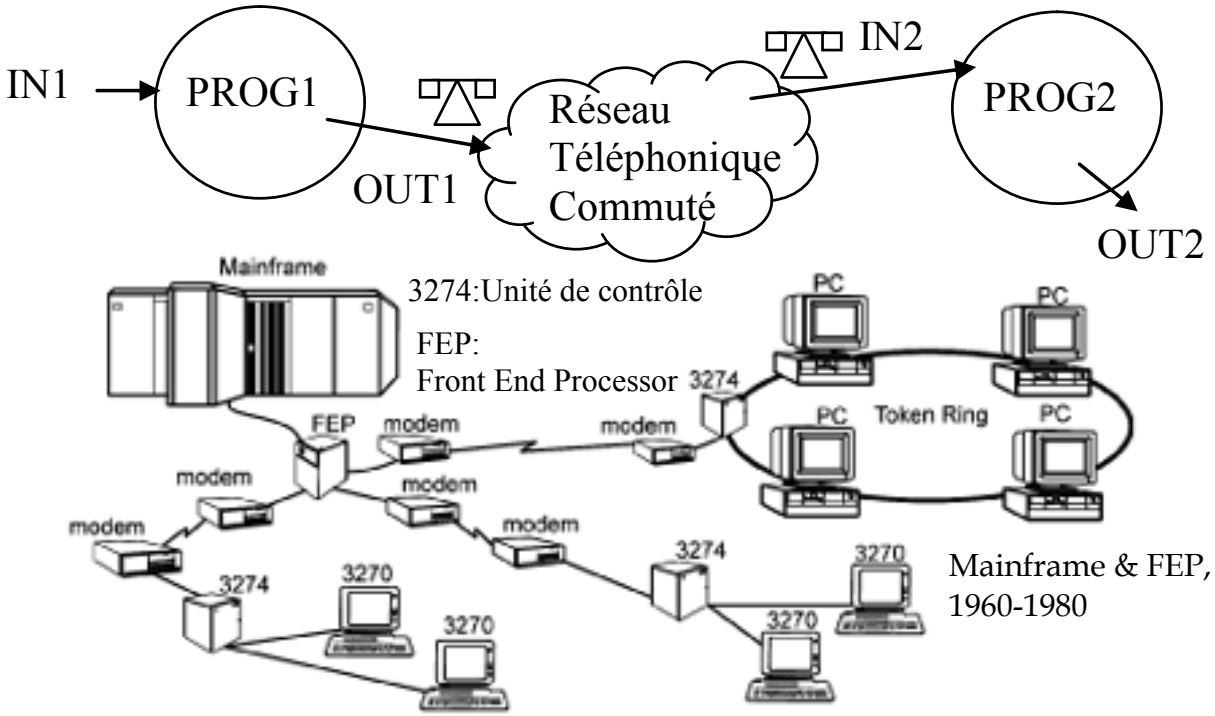


Le format ELF (*Executable Linking Format*), utilisé par de nombreux systèmes UNIX, (et Windows) est un exemple de l'architecture dite de *Von Neumann* dans laquelle le code et les données sont logés dans une mémoire commune. Il existe un autre concept, l'architecture d'*Harvard*, dans lequel le code 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 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 surs (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. Sous UNIX des applications telles que RLOGIN permettent de se connecter à une machine distante à l'aide d'un identifiant et d'un mot de passe, qui sont échangés sans protection particulière.

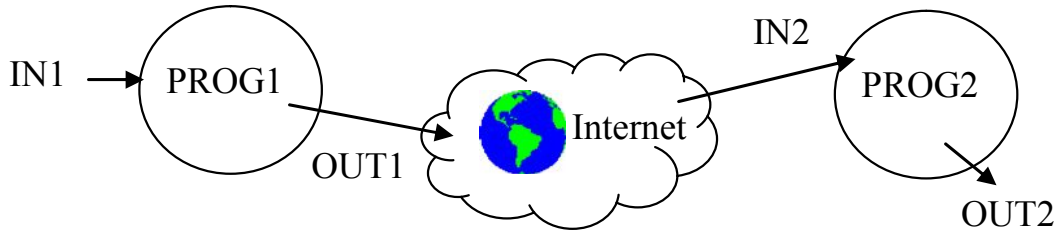


Mainframe & FEP, 1960-1980

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 hétérogènes, le réseau n'est pas sûr, le nombre d'utilisateurs est important.

D'abord absente des premiers standards, la sécurité de l'Internet (*Security Architecture for the Internet Protocol*, RFC 825, 1995), puis du WEB (*Secure Socket Layer*, SSL 3.0, 1996) 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.



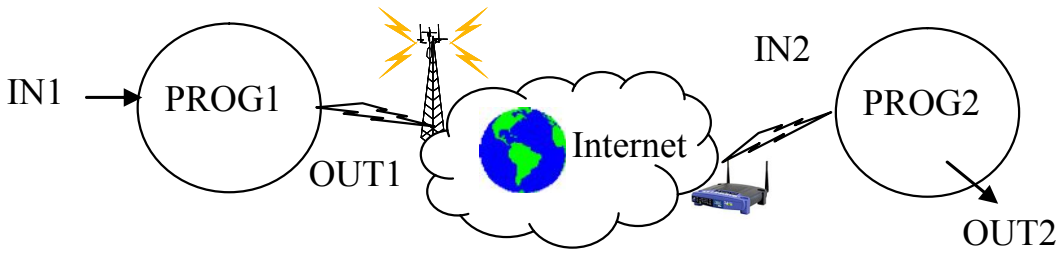
Poster de Tim Berners-Lee à la conférence *ACM Hypertext 91*.

[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, 1991

L'internet sans fil, les réseaux radio

La dernière révolution des communications repose sur les technologies de réseaux IP sans fil, tels que Wi-Fi[4], WiMAX[5] ou réseaux cellulaires 3G/4G/5G/6G. 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[6] s'affirme, le contrôle des accès réseaux et la confidentialité des données.



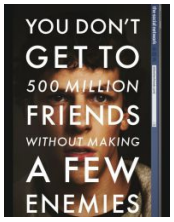
Transfert de fichiers et streaming

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'informations, en temps réel, 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, fibre optique) établit un lien point à point avec le réseau de l'*Internet Service Provider* (ISP).

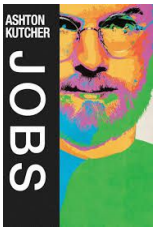
Du Data Center au Cloud



Durant la période 2000-2010 les grands fournisseurs de services de l'Internet (tels que Google, Facebook, Amazon…) 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

[4] IEEE 802.11b, Septembre 1999
[5] IEEE 802.16e, Décembre 2005
[6] IEEE 802.11i, 2004

un navigateur ou une machine virtuelle, la sécurité d'accès au service repose sur des technologies WEB.



Annoncé en 2007, l'iPhone marque le début de l'internet mobile. De par sa nature, le stockage de données sensibles dans un objet portable implique la disponibilité de composants électroniques sécurisés adaptés (perte, vol, hacking…) tels que *Secure Elements* (contrôleur NFC, carte SIM/USIM) ou *Trusted Execution Environment* (TEE).

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*). L'idée est d'extraire du protocole HTTP les principes de son efficacité.

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 les 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.

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 partagées 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 2023 sans solution éprouvée; les tentatives infructueuses de développement des techniques WBC (*White Box Cryptography*[7]) 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)

[7] https://cryptoexperts.com/technologies/white-box/

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 (mots de passe...) collectées par divers procédés.

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 maître de chiffrement de fichiers, liée à un mot de passe. Le confinement dans un espace sûr (*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 mots clés

Network Detection and Response, (NDR) est un système de sécurité de réseaux

Endpoint Detection and Response (EDR) est un système de sécurité de terminaux

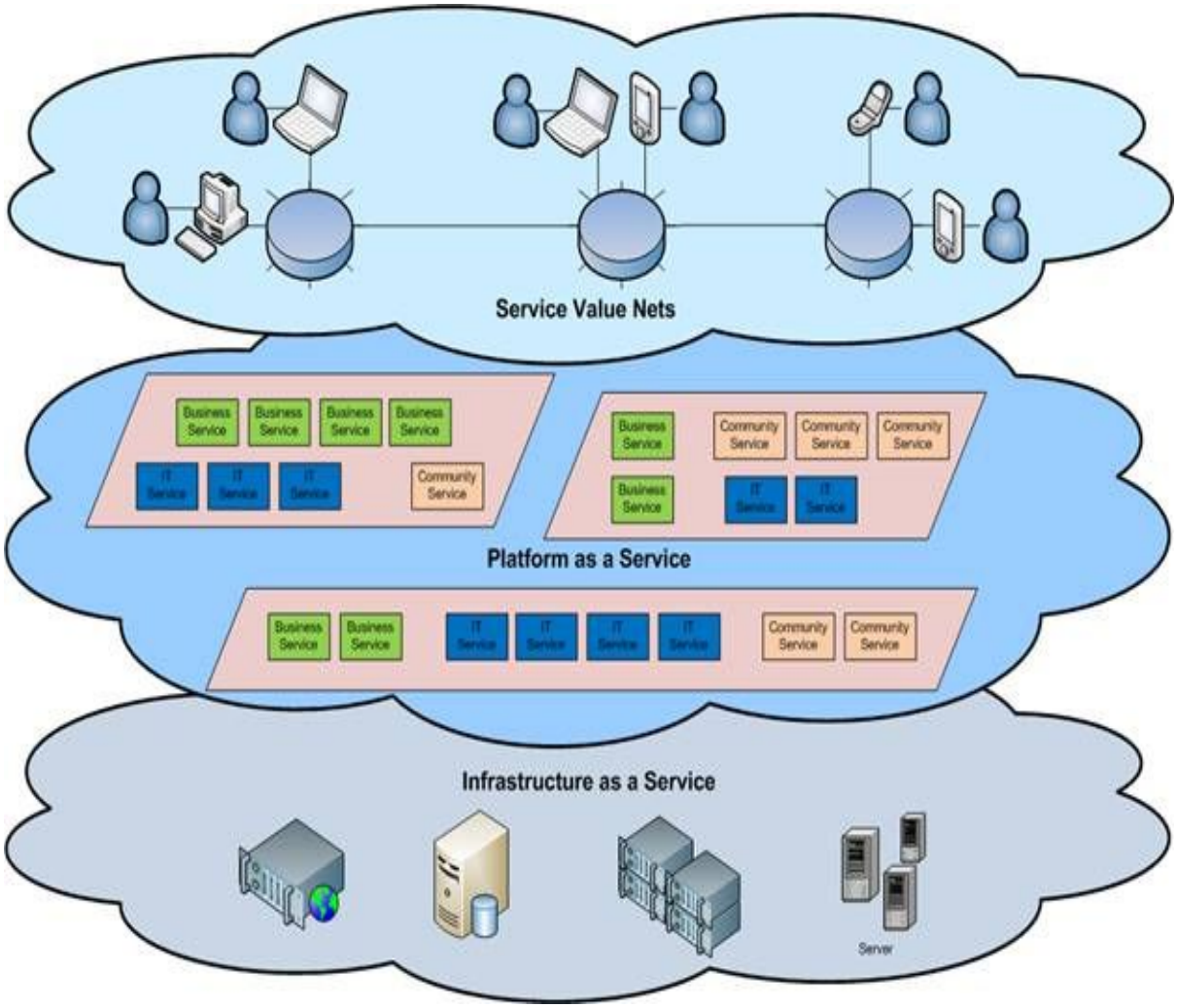
Extended Detection and Response (XDR) est un système étendu de réponse aux attaques.

Plateformes

Cloud Computing

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 (c'est la notion d'**élasticité**), 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.

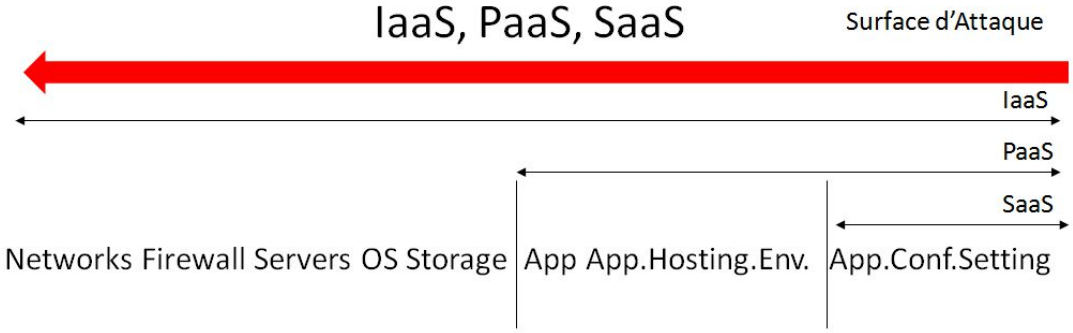
- PaaS ou *Plateform as a Service*. C'est un environnement (APIs, Machine Virtuelles[8]) permettant de construire/déployer des applications par exemple Google App Engine ou Microsoft Azure.

- 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.

[8] L'hyperviseur Xen a été créé à la fin des années 90, par Keir Fraser et Ian Pratt, dans le cadre du projet de recherche *Xenoserver* à 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 de la législation 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

– La prévention de bugs de sécurité des hyperviseurs,

– La détection de canaux cachés (*side channel*) inter VM

- Le matériel sécurisé, hyperviseur certifié (Critères Commun - CC), Unix durci (SE Linux…).

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 sécurisées 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.

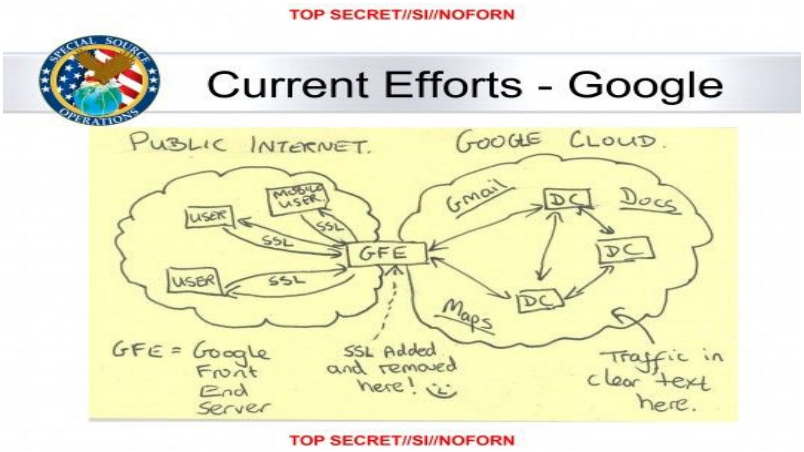
– Login/password,

- One Time Password (OTP),

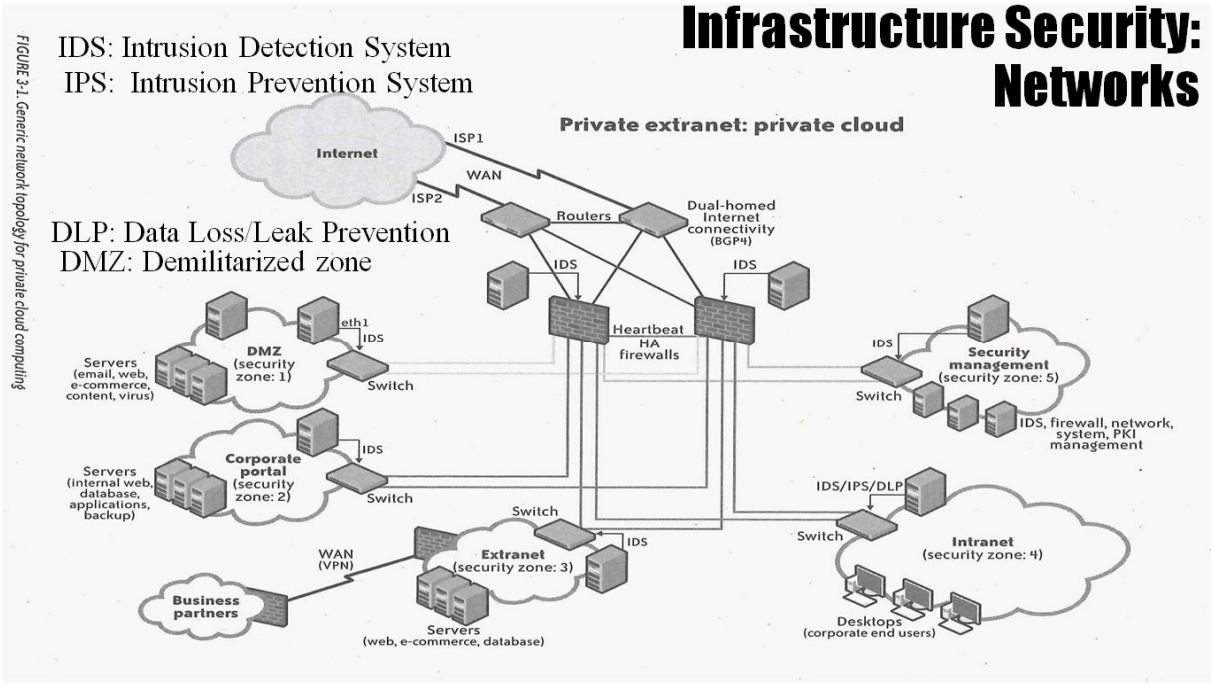
– Certificats,

– Stockage sécurisé de clés,

–Dispositifs *Hardware Secure Module* (HSM).

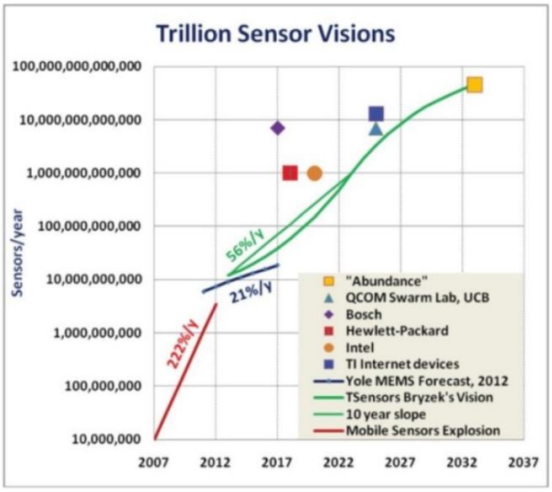


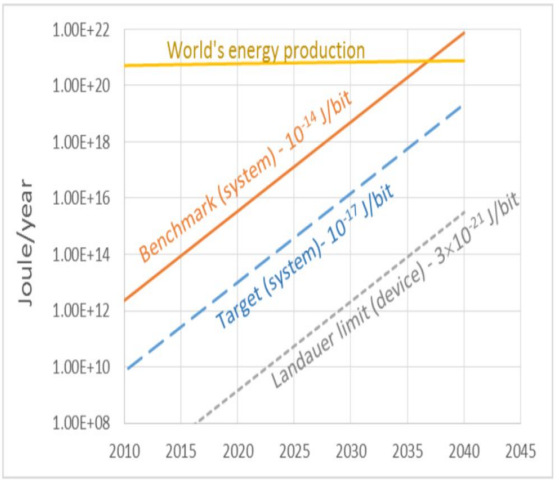
En 2013 les échanges entre les data center de Google n'étaient pas chiffrés (GFE= Google Front End server, DC= Data Center)



Sécurité des infrastructures

Internet of Things (IoT)





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

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

Quelle est l'énergie nécessaire à la commutation d'un bit ?. Soit N électrons (de charge q =1,6 10^{-19} Coulomb) l'énergie (en Joule) dissipée sous un volt est de ½ NqV. Pour 1 volt la valeur de 10^{-14} J/bit correspond à 1,25 million d'électrons stockés dans les capacités des transistors CMOS. Soit encore 100 Watts (10^{16} x 10^{-14}) pour 10^{16} commutations par seconde de 10 millions de bits (10^7) à la fréquence 1 GHz (10^9).

Le moteur de recherches SHODAN (https://www.shodan.io/) recense les dispositifs connectés. De nombreuses attaques/intrusions s'appuient sur l'usage de mots de passe par défaut.

TOP 10 des attaques IoT, OWASP 2018

En 2018 la fondation OWASP a publié une liste des 10 vulnérabilités les plus courantes de l'internet des objets.

1) Mots de passe faibles ou non modifiables

[9] "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

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

Utilisation de mots de passe ("*credential*") cassables par des attaques brute force, non modifiables ou publiés, associés à portes dérobées dans le micro logiciel embarqué ou le logiciel client, qui permettent un accès non autorisé aux systèmes déployés (voir par exemple https://www.shodan.io)

2) Services réseau non sécurisés

Services réseau non utilisés ou non sécurisés s'exécutant sur un objet connecté, qui compromettent la confidentialité, l'intégrité, l'authenticité ou la disponibilité de ses données, ou qui permettent un contrôle à distance non autorisé.

3) Interfaces non sécurisées

Interfaces WEB, *backend API*, cloud ou mobiles non sécurisées interagissant avec l'objet, et permettant sa compromission ou celle des composants associés. Les faiblesses observées sont un manque d'authentification/autorisation, un chiffrement absent ou faible, l'absence de filtrage des entrées/sorties.

4) Absence de mécanisme de mises à jour sécurisées

Par exemple absence de validation du micrologiciel sur l'appareil, procédure de livraison non sécurisée ("*supply chain attack*"), possibilité de rejeu de versions antérieures (*roolback*), et manque de notifications des mises à jour.

5) Utilisation de composants non sécurisés ou obsolètes

Utilisation de composants de logiciels obsolètes ou non sécurisés. Modification du système d'exploitation, utilisation de logiciels ou de composants matériels tiers provenant d'une chaîne de distribution compromise.

6) Protection insuffisante de la vie privée

Les informations personnelles de l'utilisateur stockées sur l'objet ou dans l'écosystème associé, sont utilisées de manière non sécurisée, abusive ou sans autorisation.

7) Transfert et stockage de données non sécurisés

Absence de chiffrement ou de contrôle d'accès aux données sensibles stockées dans l'écosystème, durant le transport ou pendant le traitement.

8) Manque d'administration des objets.

Absence d'administration de la sécurité sur les objets déployés, y compris la gestion du parc, la gestion des mises à jour, la désactivation, la surveillance des systèmes.

9) Paramètres par défaut non sécurisés

Les objets ou les systèmes sont livrés avec des paramètres par défaut non sécurisés, ou n'ont pas la capacité d'améliorer la sécurité en interdisant aux opérateurs de modifier les configurations.

10) Manque de durcissement hardware

Absence de mesures de durcissement hardware (*tamper resistant*), permettant aux attaquants d'obtenir des informations sensibles, ou de prendre le contrôle local de l'objet.

Protocoles de l'Internet of Things (IoT)

La RFC 7228 ("*Terminology for Constrained-Node Networks*", 2014) propose une 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 (TCP-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 classiques (notebook, tablettes,…).

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. CoAP peut être compris comme un proxy http, par exemple entre IPv4 et IPv6.

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 supporte 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.

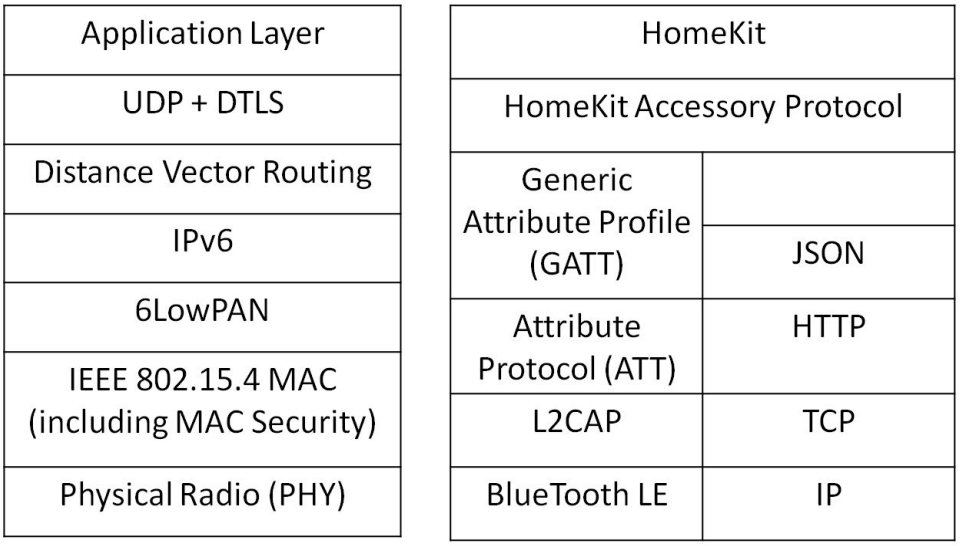
MQTT (MQ Telemetry Transport)

MQTT[11] (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.

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).

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



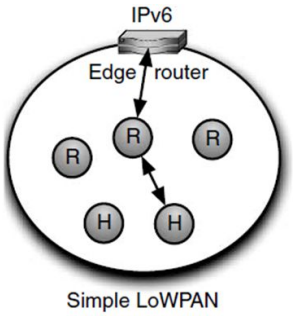
Pile logicielle 6LowPAN (à gauche) et Apple HAP (à droite)

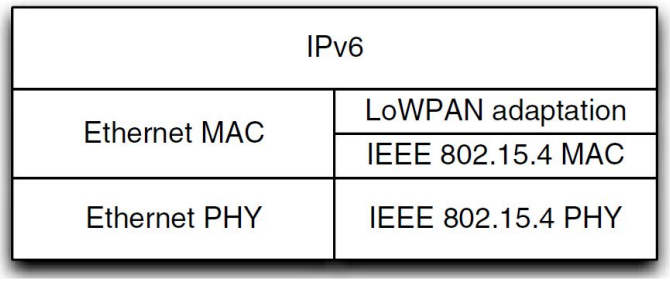
Un objet se décompose généralement en quatre entités logiques et matérielles: 1) un circuit électronique (board) qui comporte typiquement un processeur, 2) un SOC (*System On Chip*) radio, 3) un système d'exploitation ou un bootloader, 4) une pile de communication et un environnement (*framework)* applicatif. La syntaxe des messages d'application s'appuie sur JSON (*JavaScript Object Notation*), un format d'échange de données (*Data Interchange Format*) en mode texte. La structure d'un arbre JSON est décrite par un schéma JSON. Un document JSON comprend deux types d'éléments structurels : des ensembles (objets) de paires clé (key), valeur (value), et des listes ordonnées de valeurs (tableaux). 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 classique, 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).

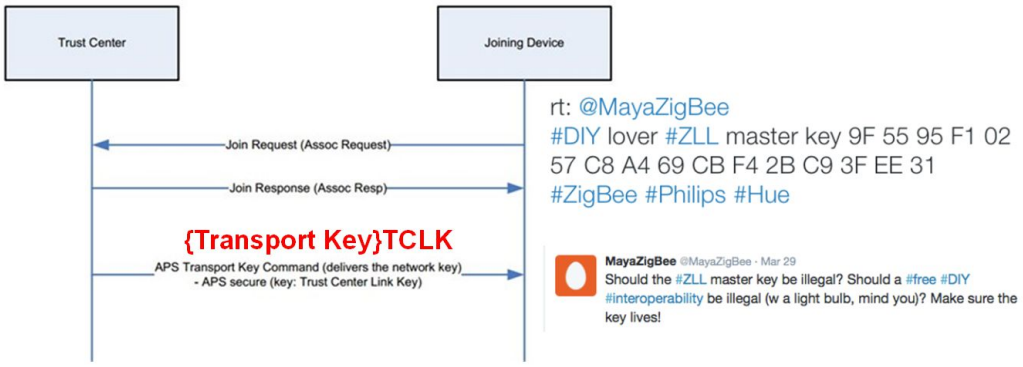
De surcroit des mécanismes de contrôle d'accès utilisant des ACLs (*Access Control List*) ou des jetons (*OAuth 2.0 Authentication*), peuvent être implémentés dans la couche applicative.





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[12] (*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 des modes 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).

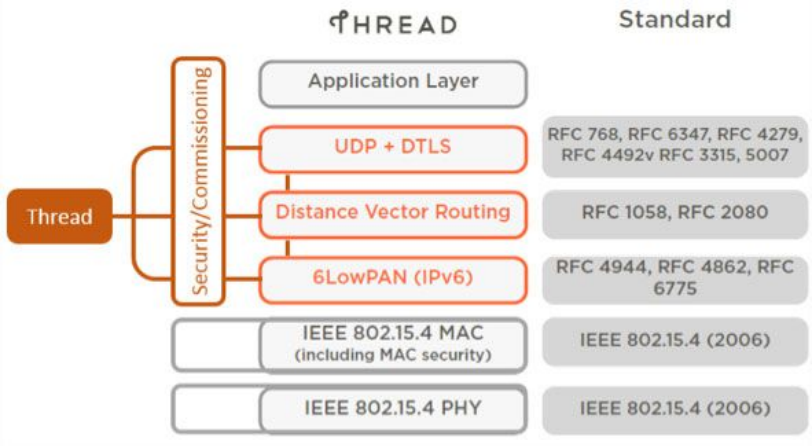
Le protocole IPSEC est bien sur compatible avec cet environnement. Cependant ZigBee conformément au standard IEEE 802.15.4 dispose d'algorithmes de chiffrement de trame, typiquement selon le mode AES-CCM 128 bits, et possède un jeu de trois clés. Un secret à long terme (*Master Key*) partagé par tous 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), et chiffrée grâce au *Trust Center Link Key*. La clé réseau (*Network Key*) est commune à tous les participants et chiffre les trames émises en diffusion (*broadcast*).



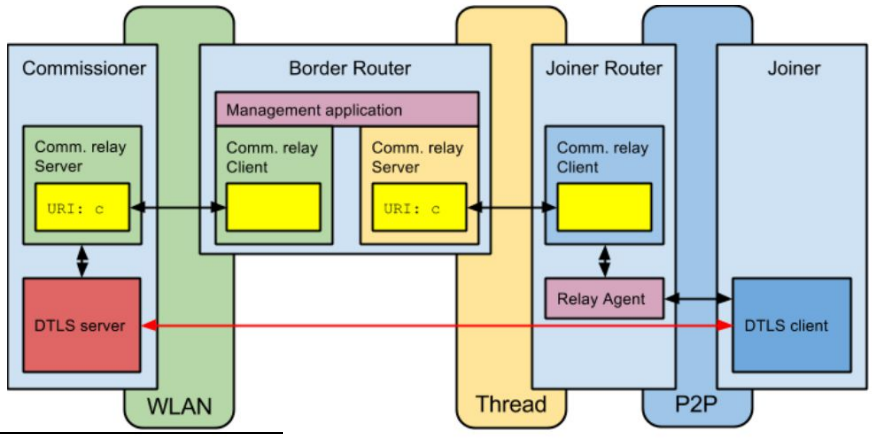
[12] 6LoWPAN: The Wireless Embedded Internet, Zach Shelby, Carsten Bormann, Wiley, 2009

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 l'aide d'échanges Diffie-Hellmann sur des courbes elliptiques, et de signature de Schnorr. Selon Thread un objet (*joiner*) s'identifie auprès 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 informations protégées par un mécanisme KEK (*Key Establishment Key*).

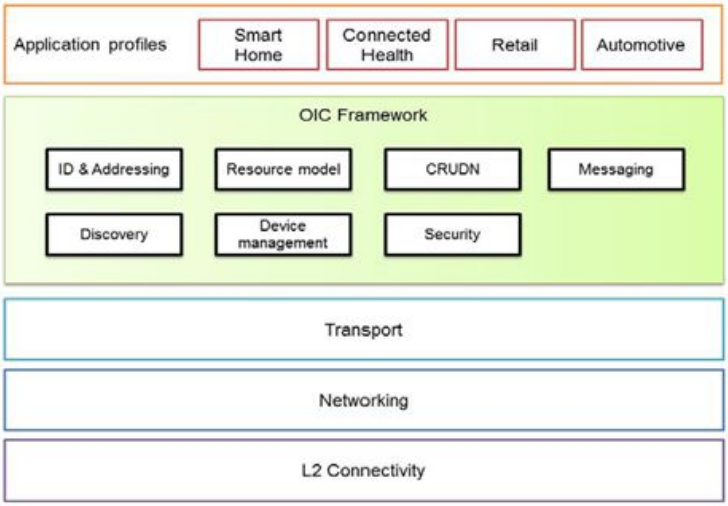


Thread est basé sur le système d'exploitation LINUX. Des sociétés telles que Silicon Labs (www.silabs.com) proposent des cartes mères de développement. Le thermostat NEST implémente les concepts Thread. L'article[13] met en évidence l'absence de Secure Boot et présente une attaque permettant de démarrer le système à partir d'une clé USB, puis de modifier ses fonctionnalités.



[13] "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 symétrique (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) renseignant 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 (pile BlueTooth). 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[14] 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 une clé maître. Un contrôleur délivre aux nœuds du réseau une clé de ligne (*Link Key*) chiffrée à l'aide du *MasterSecret*. Il 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[15] 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[16] 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 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[17] 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 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

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

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

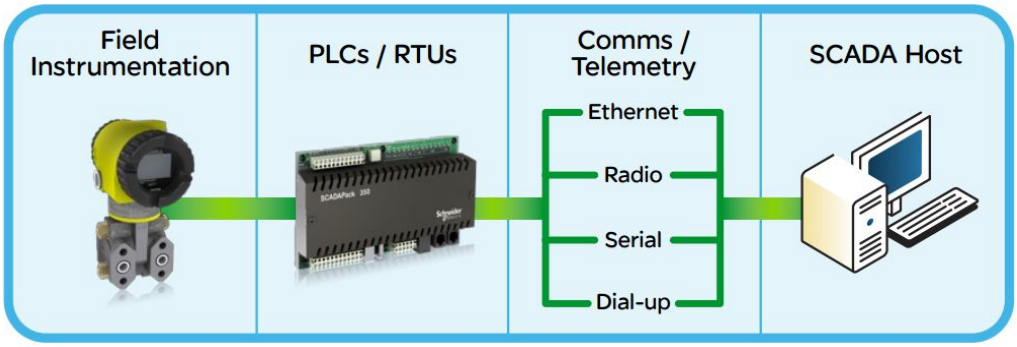
[16] A Light Bulb Worm, Details of the Philips Hue Smart Lighting Design, Colin O'Flynn - August 1, 2016.

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

REST transporté par les protocoles HTTPS ou XMPP, avec des jetons d'autorisation OAuth 2.0 (RFC 6749) délivrés par des serveurs d'authentification (AS) Google.

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[18] [19] (*Supervisory Control And Data Acquisition*), comporte des dispositifs de contrôle tels que PLC (*programmable logic controller*) ou RTU (*remote telemetry unit*) communiquant via les protocoles MODBUS ou DNP3, fonctionnant en mode Master/Slave (sur des liaisons séries RS-232, RS-485) ou Client/Server (TCP/IP).

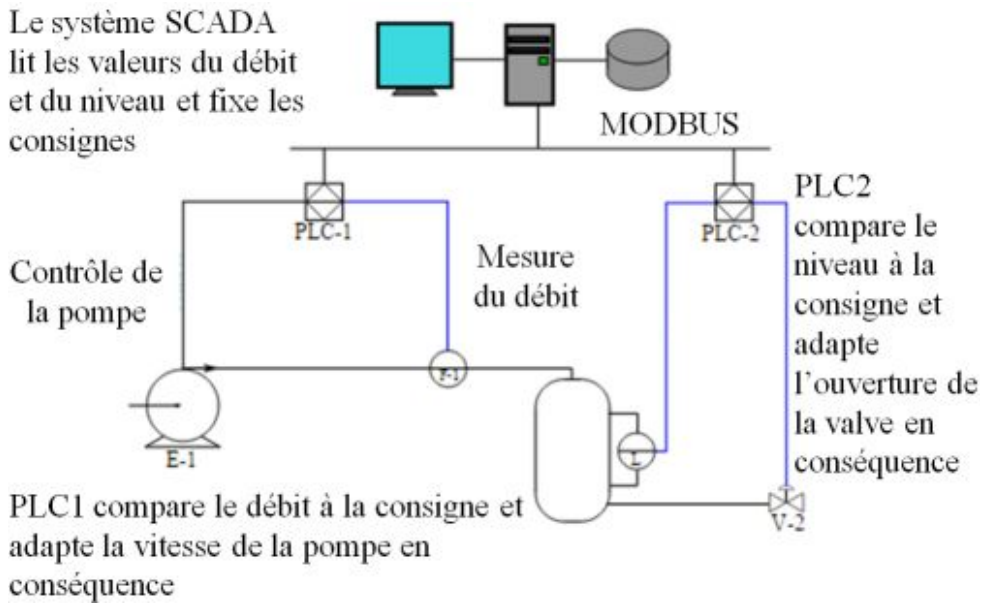


START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 octet	2 octets	2 octets	N octets	2 octets	CR LF

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

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

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



Exemple de système SCADA.

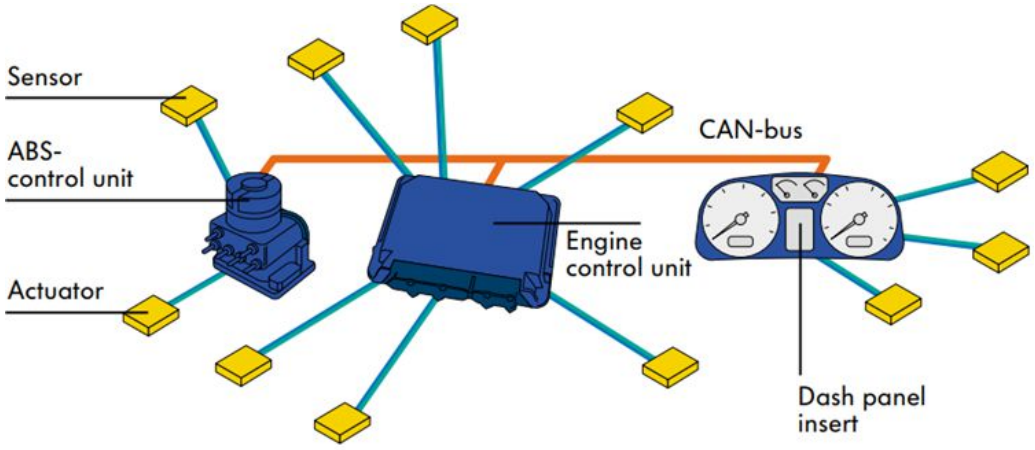
STUXNET, une arme logicielle

Le vers informatique *Stuxnet* découvert en 2010 a permis[20] la destruction de machines industrielles (un millier de centrifugeuses...) en attaquant la 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.

CAN Bus.

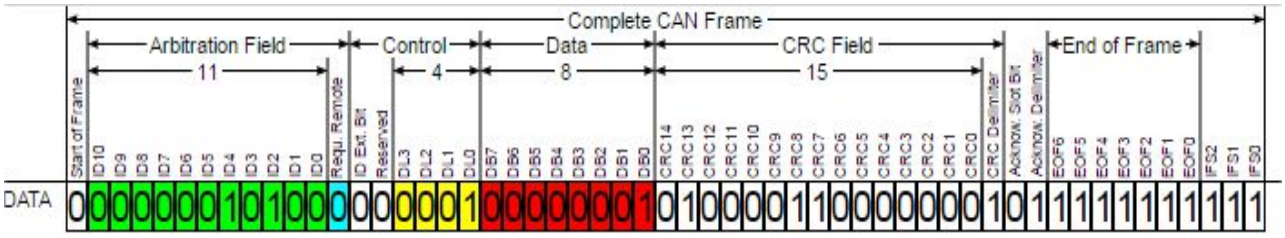
Une automobile moderne comporte un ensemble de modules électroniques (une cinquantaine) dénommés *Electronic Control Units* (ECUs) connectés en réseau et destinés au contrôle et à la gestion du véhicule (freins, direction, pneus, ...).

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



Exemple de CAN-bus reliant trois ECUs

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



Un paquet CAN Data (1 octet de donnée)

Un paquet CAN comporte un identifiant (11 ou 29 bits, mais 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 leur identifiant. Le débit usuel du CAN bus est de 500 Kbit/s (soit 2µs par bit).

Les deux types principaux de paquets CAN sont les trames DATA, et les trames REQUEST. Ces dernières sont une requête d'émission pour un ID particulier, le champ longueur indique dans ce cas la taille des données attendues, et le bit RTR (*Remote Transmission Request*) est positionné.

Les données sont codées selon un format propriétaire ou en conformité avec des standards tels que ISO-TP ou OBD-II (*On Board Diagnostic*s). Des paquets de diagnostic sont utilisés uniquement à des fins de maintenance.

Conformément aux normes ISO-TP (ISO 15765-2) 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 (pas de chiffrement ni contrôle d'intégrité).

CAN-TP Header

Bit offset	7 .. 4 (byte 0)	3 .. 0 (byte 0)	15 .. 8 (byte 1)	23..16 (byte 2)	....
Single	0	size (0..7)	Data A	Data B	Data C
First	1	size (8..4095)		Data A	Data B
Consecutive	2	index (0..15)	Data A	Data B	Data C
Flow	3	FC flag (0,1,2)	Block size	ST	

Header CAN-TP

Single Frame (SF)

Byte 0		Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Bits 7 - 4 Frame Type (0 for SF)	Bits 3 - 0 # of data bytes in this message	Service ID (for UDS)	Data	Data	Data	Data	Data	Data

Single Frame selon le standard ISO_TP

La norme OBD-II (standard SAE J/1979) date des années 1990; elle est obligatoire en Californie depuis 1996. Elle permet la lecture des *Diagnostic Trouble Codes* (DTC) standardisés ou propriétaires, ainsi que les informations temps réel en provenance de capteurs connectés aux calculateurs de bord. Elle s'appuie sur des trames CAN et le standard ISO-TP. Le protocole comporte des messages de requête et de réponse. Le premier octet d'une requête indique par le mode (01 par exemple) et le deuxième l'identifiant de protocole (PID). En cas de succès la réponse débute par l'octet mode+0x40, suivi de l'octet PID, et de données. La collecte du régime moteur (tr/mn) est illustrée ci dessus:

Requête: 7DF 08 02 01 0C [6 padding bytes]

Réponse: 7E8 08 04 41 0C 11 42 [3 padding bytes]

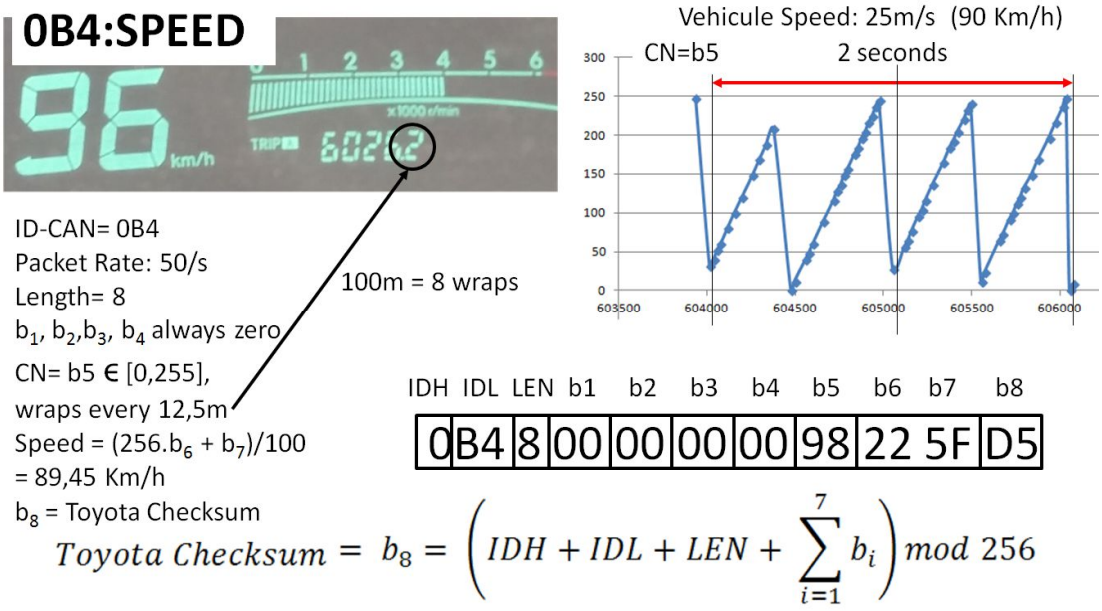
Le CAN-ID 7DF est une adresse de diffusion, le mode 01 signifie "*Show current data*" , le PID 0C désigne le régime moteur.

L'ECU qui possède l'information (CAN-ID = 7E8) délivre une réponse avec le mode 41 (0x40 + 1) et le PID de la requête, les deux derniers représentent l'information demandée.

Car Hacking

L'article[21] 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.

[21] Chris Valasek, Charlie Miller "Adventures in Automotive Networks and Control Units", 2014



Exemple d'injection de paquet CAN depuis une prise OBD-II.

En 2015 l'article[22] 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 pour le 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.

L'article[23] détaille des attaques réalisées à l'aide d'une sonde CAN réalisée à partir d'un Arduino et d'un contrôleur CAN bus MCP2515.

[22] Charlie Miller, Chris Valasek "Remote Exploitation of an Unaltered Passenger Vehicle", 2015

[23] Pascal Urien, "Designing Attacks Against Automotive Control Area Network Bus and Electronic Control Units", 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)

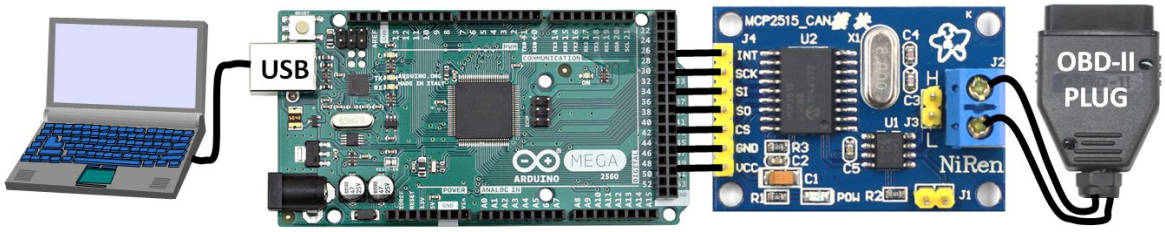
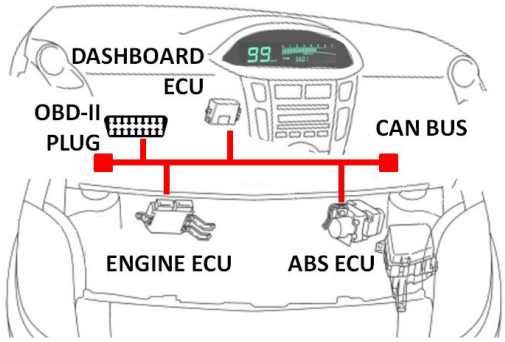


Illustration d'une sonde CAN



Cette sonde permet d'afficher des informations erronées sur le tableau de bord (vitesse,…), de modifier le compteur kilométrique, de couper l'injection de carburant dans le moteur (vitesse < 20Km/h), de différer le freinage de 1,5s pour de faible vitesse (< 2Km/h), et d'initialiser la programmation de l'ECU (moteur à l'arrêt) ce qui empêche le véhicule de démarrer.

De manière plus générale le rapport[24] 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.*"

Surface d'attaque du véhicule connecté autonome

Un véhicule de nouvelle génération propose un ensemble de fonctionnalités, telles que:

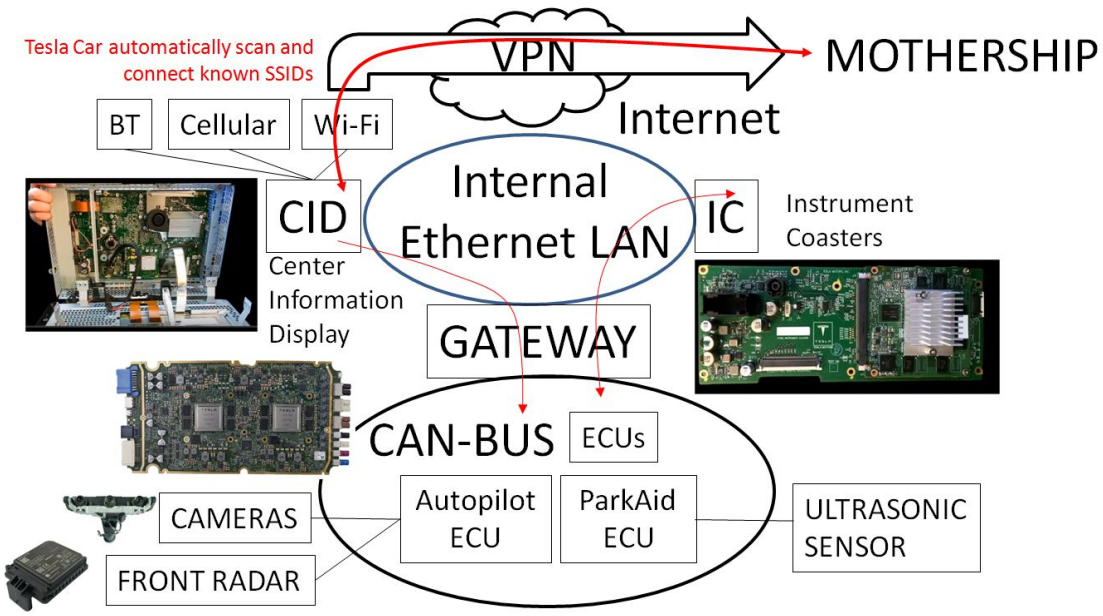
- Contrôle du véhicule via les messages CAN
- Contrôle d'accès physique (clés)
- Diagnostiques OBDII
- Recharge des batteries
- Mises à jour logicielles
- Advanced Driver Assistance System – ADAS
- Conduite autonome (SAE J3016) : 6 niveaux de 0 à 5
- V2X services (V2C Vehicule to Vehicule, V2I Vehicule to Infrastructure)

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

Un véhicule de nouvelle génération possède de multiples interfaces réseaux:

- CAN Bus & ECU (Electronic Control Units)
- LAN (Local Area Network): Ethernet
- WLAN (Wireless Area Network): Wi-Fi
- Dedicated Short-Range Communications (DSRC) pour les services V2X
- LPAN (Local Area Network): Bluetooth
- Radio Low Frequency (~150KHz) et Ultra High Frequency (~400MHz)
- LTE (Long Term Evolution) 4G, 5G
- Internet
- GPS (Global Positioning System)

Exemple d'architecture de communication d'une TESLA



L'infrastructure de communication d'une TELSA se décompose schématiquement en trois niveaux:

- Le réseau internet et le data center du constructeur (Mothership) qui gère la flotte de véhicule. Chaque véhicule, identifié par un VIN (*Vehicule Identification Number*), possède un certificat, qui permet d'établir au terme d'une procédure de mutuelle authentification un VPN avec les serveurs distants)

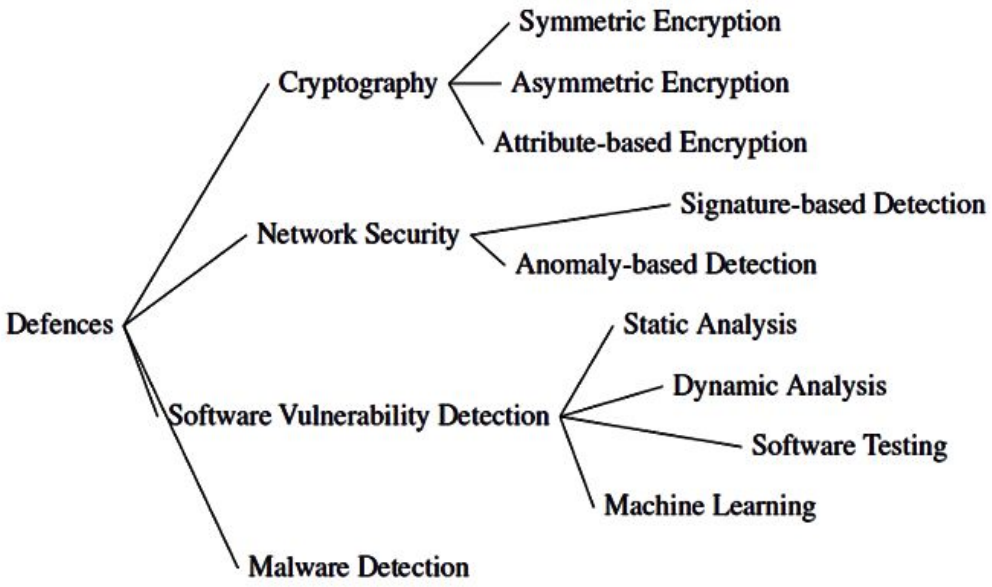
- Un réseau local Ethernet *Infotainment* (*Information* +*Entertainment*). Ce dernier gère des interfaces pour Internet (Wi-Fi, réseau cellulaire), Bluetooth et CAN-Bus. Il comporte trois nœuds, CID (Center Information Display, Unix), IC (Instrument Coaster, Unix), et passerelle CAN-BUS (RTOS).

- Un réseau CAN-BUS. Un système *Autopilot* réalise les services *Full Self Driving* (FSD); ce dernier directement connecté aux radars et cameras; il gère également un lien Ethernet Gigabit avec la passerelle.

Types d'attaques[25]

Level	Exposures	TYPE OF ACCESS		IMPACT POTENTIAL		
		Physical access	Wireless access	Safety	Data Privacy	Car-jacking
HIGH	OBD II port	✓		✓		
	Wi-Fi		✓	✓		
	Cellular connection (3G/4G)		✓	✓		
	Over-the-air update		✓	✓		
	Infotainment System		✓	✓		
	Smart-phone	✓		✓		
MEDIUM	Bluetooth		✓	✓		
	Remote Link Type App		✓	✓		
	KeyFobs and Immobilizers		✓			✓
	USB	✓		✓		
	ADAS System		✓	✓		
	DSRC-based receiver (V2X)		✓	✓		
LOW	DAB Radio		✓	✓		
	TPMS		✓		✓	
	GPS		✓		✓	
	eCall		✓	✓		
	EV Charging port	✓		✓		
	CD/DVD player	✓		✓		

Méthodes de défense[26]



[25] Adi Karahasanovic & All, "Adapting Threat Modeling Methods for the Automotive Industry" - 15th ESCAR Conference, Berlin 2017

[26] Mahdi Dibaei & All, "Attacks and defences on intelligent connected vehicles: A survey", 2020

Crypto-Monnaies, Blockchain, Smart Contracts

La blockchain est un concept très novateur qui permet de partager des informations sans tiers de confiance. Elle s'appuie sur un réseau P2P utilisant des protocoles dédiés. Des transactions signées sont collectées par les nœuds du réseau, qui vérifient leur authenticité et leur cohérence, et les rassemblent dans un bloc. Ce bloc est lié au précédent selon un mécanisme de consensus tel PoW (Proof of Work) ou PoS (Proof of Stake).

Bitcoin

Les principes de la crypto monnaie Bitcoin[27] ont été définis en 2008 par l'article[28] 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 (*issuer*) qui délivre une autorisation.

Dans un contexte blockchain bitcoin Bob possède une clé privée (un nombre de 32 octets -x-) et un identifiant (dénommé adresse, @bob) dérivé de sa clé publique (sur la courbe elliptique secp256k1, g^x). Les transactions sont publiques et enregistrées/publiées dans un livre de compte (*ledger*) instancié par une série de blocs (la blockchain). Le crédit de @Bob et les transactions sont publiques et certifiées par la blockchain. Un avoir de @Bob est identifié par un montant, le hash (identifiant) d'une précédente transaction de versement, nommé UTXO (*Unspent Transaction Output*), et l'index du versement dans cette transaction. Bob réalise un transfert d'UTXO au bénéfice d'Alice identifiée par son adresse @Alice; il peut (doit) attribuer un pourboire (f*ee*) au mineur, qui est déduit de l'UTXO. Il signe la transaction et la transfère au système blockchain.

Toutes les 10 minutes (en moyenne) les transactions sont assemblées dans un nouveau bloc (soit 144 blocs par jour environ). Un entête bloc (*header*) comporte entre autre 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 :

---

[27] Andreas M. Antonopoulos, "Mastering Bitcoin", O'REILLY, 2015

[28] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System.", 2008

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

L'entropie du calcul est voisine de $32+\log_2(difficulty)$, soit 77 bits en 2022.

La difficulté est ajustée tous les 2016 blocs, soit environ toutes les deux semaines (2016/144 = 14).

La probabilité Δp de miner un bloc en temps Δt selon une difficulté D (nombre de calculs) et un hash rate h(t) (nombre de calculs par seconde) s'écrit:

$$\Delta p = \Delta t \times h(t)/D$$

Il en résulte que la densité de probabilité de minage d'un bloc est une fonction exponentielle:

$$\rho(t) = \lambda e^{-\lambda t} \qquad \lambda = h(t)/D$$

En particulier la probabilité de miner un bloc en un temps inférieur à t s'écrit

$$p(t) = 1 - e^{-\lambda t}$$

Cette relation illustre les attaques à 51% dans lesquelles deux puissances de minage s'affrontent λ1 = h1(t)/D et λ2= h2(t)/D, si λ1 est associée à la puissance de minage honnête, on obtient pour λ1 >= λ2, p1(t) >= p2(t).

Un mineur utilise une machine spécialisée (le *rig*) pour accomplir cette opération. En 2022 le système ASIC AntMiner S17 réalise 73 TH/s (73000 milliards de calculs H(x) par seconde), pour un prix de 2000$, et une consommation électrique de 2920 Watts, soit environ 0,04 Joule/Gh/s

Une difficulté de 2^{77} (en avril 2023) toutes les 10 minutes (#2^9 secondes) implique un hash rate d'environ 350 10^6 TH/s (2^{69}). La consommation électrique journalière est de l'ordre de 0,34 TWh (123 TWh / an), la production française d'énergie est de 540 TWh / an) pour un cout de 51 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 6,25 BTC en 2021, soit 900 BTC par jour, 27 Million€ pour un cours de 30000 $/BTC. Le parc de rig est d'environ 4,8 million d'unités.

La première transaction d'un bloc (le c*oinbase*) indique l'adresse à laquelle est versée la prime de minage.

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).

```
01000000 // Version
01 // number of inputs
 DE2D211EF429909B0AB8D2E7D25826A0 //TransactionID
 EDD6281EC6DEDF2B822CE5014A349E72
 01000000 // index
 8A // length of the signature Script
  47 // ECDSA Signature length
   30 44 // Sequence of (r,s) integer values
    02 20 // integer r value
     0772ABD5D37D0CAAB881DBC8912628F9
     3461839CC8D4BC007A355831A6061ED7
    02 20 // integer s value
     4CCCC34B34A9075FC09C9777EAB7A6F5
     612DA2130C1FF1C0E376AD9B2209D51D
  01 41 // Public key length
   04 // uncompressed format
   CFD7A542B8C823992AF51DA828E1B693
   CC5AB64F0CACF0F80C31A1ECA471786E
   285BDD3F1FE0A006BD70567885EF57EB
   149C8880CB9D5AF304182AC942E176CC
FFFFFFFF // sequence
```

```
01 // number of outputs
 D418040000000000 // amount in BTC
 19 // Public Key Script
  76 // OP_DUP
  A9 // OP_HASH160
  14 // hash160 length
   CB643DD608FB5C323A4A6342C1A6AC8048B409EB
  88 // OP_EQUALVERIFY
  AC // OP_CHECKSIG
00000000 // Locktime
```

The *pay-to-pubkey-hash* script is defined as:
OP_DUP [76] OP_HASH160 [A9]
<length=14> <hash160>
OP_EQUALVERIFY[88] OP_CHECKSIG[AC]

Exemple d'une transaction Bitcoin

Ethereum et smart contract

Le concept de la blockchain Ethereum a été introduit[29] 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=3500 $ en avril 2022), 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[30] 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*.

De manière similaire au bitcoin les utilisateurs sont identifiés par une adresse dérivée d'une clé publique (courbe elliptique secp256k1), et possèdent une clé privée utilisée pour la signature des transactions. Les blocs sont minés toutes les 14,0 secondes, gratifiés d'une prime (reward) de 2 Ethers (auparavant 5 puis 3 ethers), à laquelle s'ajoute la notion d'oncle (une prime de (7-rang) x reward/8 d'Ether, avec rang >=0). La production de monnaie est donc constante (6172 blocs/jour, 12,344 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 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 (1100 TH/s en 2021). Une

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

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

machine Radeon RX 480, d'un prix de 540$, consomme 165W et produit 30MH/s; soit une consommation électrique totale journalière de 145,000,000 KWh pour un cout de 21,7 Million€.

```
F8 6B  // list length= 107 bytes
80 // nonce = null (zero value)
85 04E3B29200 // gasPrice= 21,000,000,000 Wei)
82 9C40 // gasLimit= 40,000 Wei
94 777A07BAB1C119D74545B82A8BE72BEAFF4D447B //Recipient
87 2386F26FC10000 // value= 10,000,000,000,000,000 Wei
80 // data = null
1C // signature recovery parameter = 28
A0 F1DD7D3B245D75368B467B06CAD61002 // r value
   67031935B7474ACB5C74FE7D8C904097 // 32 bytes
A0 772D65407480D7C45C7E22F84211CB1A // s value
    DF9B3F36046A2F93149135CADBB9385D // 32 bytes
```

Exemple de transaction Ethereum

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 coût maximum de l'exécution du contrat (*Start Gas*).

Exemple de contrats: les jetons ERC20

Selon la recommandation ERC20 un jeton est associé à une adresse de contrat implémentant l'interface ERC20. Lors de la création le nombre de décimal (n) est défini, soit 1 jeton = 10^n unités.

function name() public view returns (string)
function symbol() public view returns (string)
function decimals() public view returns (uint8)
function totalSupply() public view returns (uint256)
function balanceOf(address _owner) public view returns (uint256 balance)
function transfer(address _to, uint256 _value) public returns (bool success)
function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)
function approve(address _spender, uint256 _value) public returns (bool success)
function allowance(address _owner, address _spender) public view returns (uint256 remaining)
event Transfer(address indexed _from, address indexed _to, uint256 _value)
event Approval(address indexed _owner, address indexed _spender, uint256 _value)

Les transferts de jeton sont réalisés par des procédures dédiées dont l'image binaire est insérée dans le champ data de la transaction, dont l'adresse de destination est celle du contrat ERC20, par exemple:

```
a9059cbb
0000000000000000000000005a8e2728f64618c2119a8fd78078a9ade19abe8
0000000000000000000000000000000000000000000000000de0b6b3a7640000
```

Function: transfer(address to, uint256 tokens)

MethodID: 0xa9059cbb = sha3["transfer(address,uint256)"]

```
MethodID: A9059CBB
[0]:  0000000000000000000000005a8e2728f64618c2119a8fd78078a9ade19abe8
[1]:  0000000000000000000000000000000000000000000000000de0b6b3a7640000
```

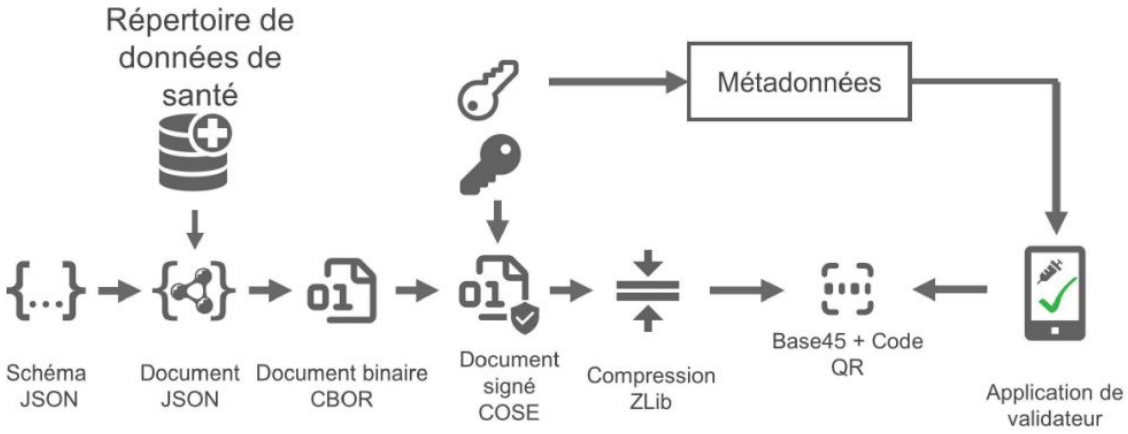
Dans cet exemple la procédure la transaction invoque la méthode *transfer()* d'un contrat ERC20, le premier paramètre est l'adresse du bénéficiaire, le deuxième paramètre est la valeur du token soit 1.000.000.000.000.000.000 (18 décimales).

Le certificat COVID numérique de l'Union Européenne

Les spécifications[31] techniques, publiées en juin 2021 par la commission Européenne, décrivent le format et les conditions d'usage d'un certificat COVID numérique.

Un certificat COVID-19 est encodé au format *Concise Binary Object Representation* (CBOR) détaillé dans le RFC 7049, et comporte une signature numérique au format *Object Signing and Encryption* (COSE) décrit par le RFC 8152. Cette structure du certificat est communément appelée "*jeton CBOR pour la toile*" (CBOR Web Token, CWT).

Le jeton CWT est compressé selon le format ZLIB, puis encodé en base45, et enfin concaténé au préfixe "HC1:". Il est généralement représenté sous forme d'un QR code, le support par un tag NFC est également possible.



Dans la RFC 8152 un champ "*Key identifier*" (kid) identifie le certificat qui contient la clé publique de l'autorité de signature, il représente les 8 premiers octets (MSB) de l'empreinte (sha256) du certificat.

Une liste de certificats est disponible à l'adresse:

https://app-static.tousanticovid.gouv.fr/json/version-34/Certs/dcc-certs.json

Chaque certificat est associé à un index, le kid exprimé en base64

Concise Binary Object Representation (CBOR)

CBOR définit 5 types de données (*major type*), associés aux trois bits de poids fort de l'octet type.

- Major type 0: entier positif ou nul, les 5 bits de poids faible définissent la valeur (de 0 à 23), 24, 25, 26, 27 définissent respectivement des entiers de 8, 16, 32, et 64 bits.

[31] https://ec.europa.eu/health/ehealth-digital-health-and-care/ehealth-and-covid-19_en

- Major type 1: entier négatif, le codage est identique à major type 0.
- Major type 2: byte string (chaine d'octets). Le codage de la longueur (un entier positif) est identique à celui de major type 0.
- Major type 3: text string (chaine de caractères au format unicode). Le codage est identique à celui de major type 2
- Major type 4: un tableau d'éléments (array, list, sequence, tuple). Le codage est identique à celui de major type 2, excepté que la longueur indique le nombre d'éléments du tableau
- Major type 5: un tableau de paires d'éléments (map, table, dictionary, hash, object). Le codage est identique au major type 2, excepté que la longueur indique le nombre de paires d'éléments, qui comprennent une clé et une donnée.

Major type 6: Tag

Major type 7: Nombre au format virgule flottante

Un parseur CBOR est disponible à l'adresse https://cbor.me/

CBOR Web Token (CWT)

Une structure CWT comporte un tag optionnel et un tableau de quatre éléments :

- Un entête protégé (*protected header*), encodé sous forme d'un octet string
- Un entête non protégé (*unprotected header*) encodé sous forme d'un *map*
- Le contenu du message encodé sous forme d'un *byte string*
- Une signature encodée sous forme d'un *byte string*

L'algorithme de signature utilisé est ECDSA sur la courbe elliptique secp256v1. L'exemple ci-dessous comporte une signature invalide

```
D284
4DA2012604487C62EEBE0EA7E709
A0
58EEA40164434E414D041A64B1380C061A60EED1C0390103A101A4617681AA626369781D757
26E3A757663693A30313A46523A414243444546474849 4A4B4C233362636F62465262646E02
6264746A323032312D30372D313462697364434E414D626D616D4F52472D313030303330323
135626D706C45552F312F32302F313532386273640262746769383430353339303036276 70
674A3037425830336 36 46F626A313736392D30382D3135636E616DA462666E684E41504F4C4
54F4E62676E69424F4E4150415254456366 6E74684E41504F4C454F4E63676E7469424F4E41
504152544563766572653 12E332E30
5840BADBADBADBADBADBADBADBADBADBADBADBADBADBADBADBADBADBADBADBADBADBADBADBA
DBADBADBADBADBADBADBADBADBADBADBADBADBADBBADBADBADBADBADB
```

Le CWT comporte un tag et un tableau de 4 éléments.

```
D2                  # tag(18)
  84                # array(4 éléments)
    4D              # bytes(longueur 13)   A2012604487C62EEBE0EA7E709
    A0              # map(0 élément)
    58 EE           # bytes( longueur 238)
    58 40           # bytes (longueur 64)
```

L'entête protégée (map à deux éléments A2) définit l'algorithme de signature (alg, clé 1) et l'identifiant de clé (Key Identifier, kid, clé 4).

```
A2          # map(2 éléments)
  01        # unsigned(1)
  26        # negative(6), ECDSA
  04        # unsigned(4)
  48        # bytes(8)  7C62EEBE0EA7E709    fGLuvg6n5wk=
```

Dans le cas du passe sanitaire le contenu est un tableau de 4 *maps*, définissant

- l'émetteur (Issuer, iss, clé 1)
- la date d'expiration (exp, clé 4), au format UNIX
- la date d'émission (Issued At, iat, clé 6), au format UNIX
- le certificat sanitaire (hcert, clé de revendication -260).

```
A4              # map(4)
  01            # unsigned(1)
  64            # text(4)
    434E414D    # "CNAM"
  04            # unsigned(4)
  1A 64B1380C   # unsigned(1689335820)
  06            # unsigned(6)
  1A 60EED1C0   # unsigned(1626264000)
  39 0103       # negative(259)
```

Le certificat (HCERT) est un tableau de maps renseignant un ensemble de paramètres, en particulier nom, prénom et date de naissance

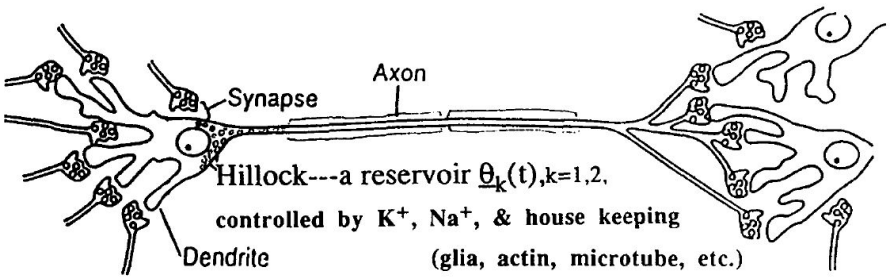
{1: {"v": [{"ci": "urn:uvci:01:FR:ABCDEFGHIJKL#3", "co": "FR", "dn": 2, "dt": "2021-07-14", "is": "CNAM", "ma": "ORG-100030215", "mp": "EU/1/20/1528", "sd": 2, "tg": "840539006", "vp": "J07BX03"}], "dob": "1769-08-15", "nam": {"fn": "NAPOLEON", "gn": "BONAPARTE", "fnt": "NAPOLEON", "gnt": "BONAPARTE"}, "ver": "1.3.0"}}

La signature est le couple (r,s) produit par la signature ECDSA sur la courbe Prime256.

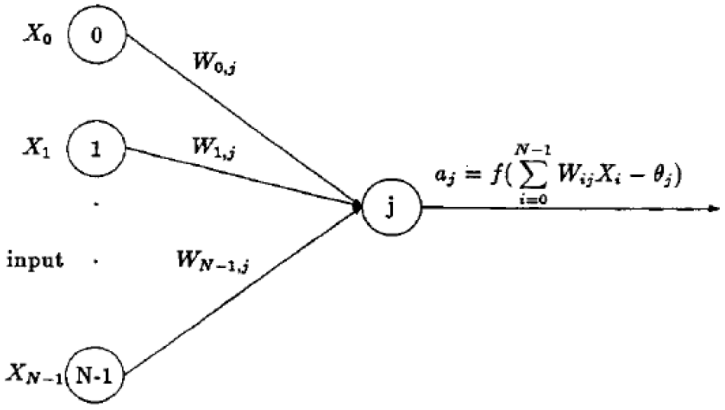
Deep Learning

Introduction aux techniques Deep Learning (DL)

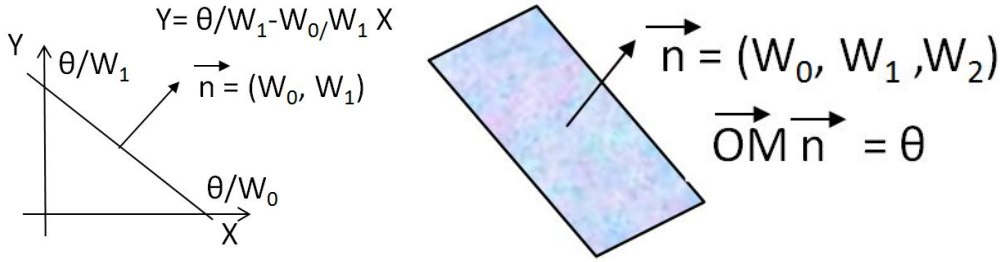
In 1943, Warren McCulloch and Walter Pitts ont conçu[32] le premier réseau de neurones, par analogie avec le cerveau humain.



L'approximation d'un neurone (j) s'écrit Yj = f(∑ Wij Xi – θj), Xi étant les entrées (n entrées), θj le seuil, et f la fonction de d'activation. La sortie de f varie entre -1/0 et 1, par exemple c'est la fonction de Heaviside H(x), ReLU(x), ou la fonction sigmoïde.

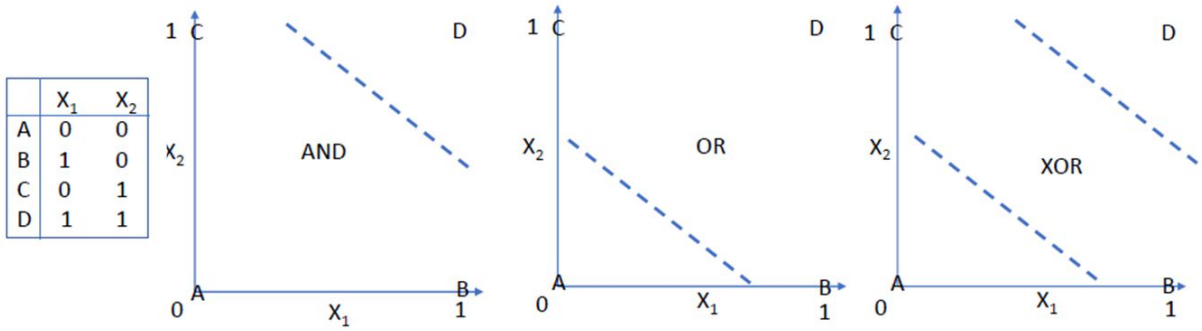


Dans un espace affine euclidien à N dimensions, le neurone définit deux zones séparées par un hyperplan (P), P:{M, $\overrightarrow{OM}\,\overrightarrow{n}= \theta$ }, $\overrightarrow{n}=(W_1,\ldots,W_{n-1})$



Le neurone de *McCulloch & Pitts* peut réaliser les fonctions AND ou OR, mais pas le ou exclusif (XOR).

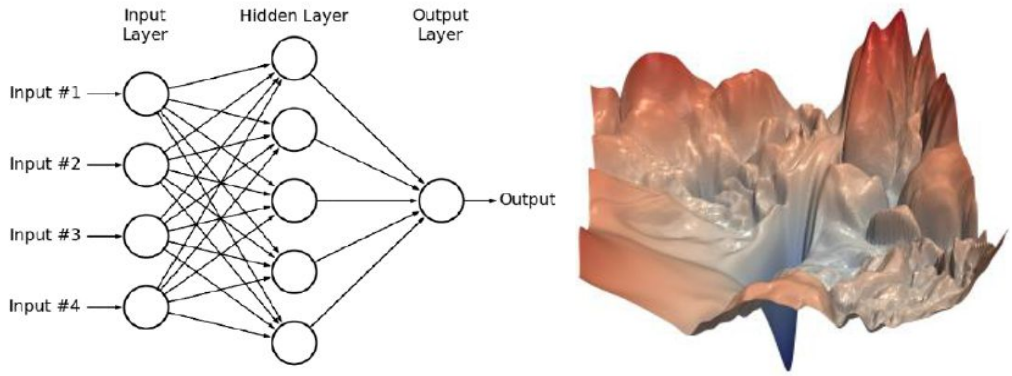
[32] Warren S. McCulloch and Walter Pitts, Walter "A logical calculus of the ideas immanent in nervous activity" The bulletin of mathematical biophysics, 5(4), 115-133, (1943)



Frank Rosenblatt a proposé[33] en 1958 un algorithme d'apprentissage supervisé (le *Perceptron*), implémenté pour la reconnaissance d'image.

En 1986, l'article[34] décrit le premier algorithme concept d'apprentissage par un algorithme de rétro propagation d'erreur (qui deviendra par la suite la méthode d'apprentissage par descente de gradient).

L'article[35] a démontré en 1989, qu'un réseau de neurones à trois couches (input layer, hidden layer, output layer) peut approximer toute fonction Y=F(X,W).



Un réseau à N couches Y=F(X,W) possède E entrées (X) et S sorties (Y) Yj= Fj(X,W) et W poids (Wk). Les données d''apprentissage (*training data*) sont un ensemble de m vecteurs (X,Y).

Un algorithme (la *descente de gradient*) produit les poids W du modèle, en minimisant les erreurs selon une fonction de coût, par exemple $\sum \sum (Yj,i - Fj(Xi,W))^2$

Les réseaux CNN (*Convolutional Neural Network*) sont inspirés des travaux David H. Hubel and Torsten N. Wiesel (prix Nobel en Physiologie et Médecine en 1981), relatifs au traitement de l'image par le cerveau humain, selon un réseau de neurones organisé en trois couches, V1, V2, V4.

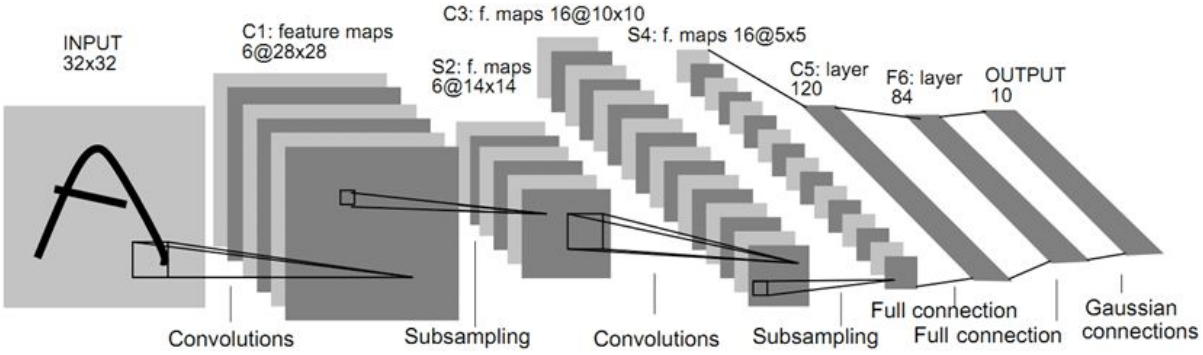
Par exemple le réseau LeNet5[36], dédié à la reconnaissance de caractères, publié en 1998, comportait 60.850 paramètres d'apprentissage et 340.918 connections.

---

[33] Frank Rosenblatt "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," Psychological Review, 65(6), 386-408. 1958

[34] Rumelhart, D.E., Hinton, G.E. and Williams, R.J. "Learning representations by back-propagating errors," Nature, 323, 533-536., 1986

[35] Kurt Hornik, Maxwell Stinchcombe, Halbert White, "Multilayer feedforward networks are universal approximators" Neural Networks, 2(5), 359-366, 1989



Le réseau LeNet5, 1998

Attaques des systèmes Deep Learning

Le modèle Y=F(X,W) est réalisé par des processeurs GPU ou des circuits intégrés dédiés.

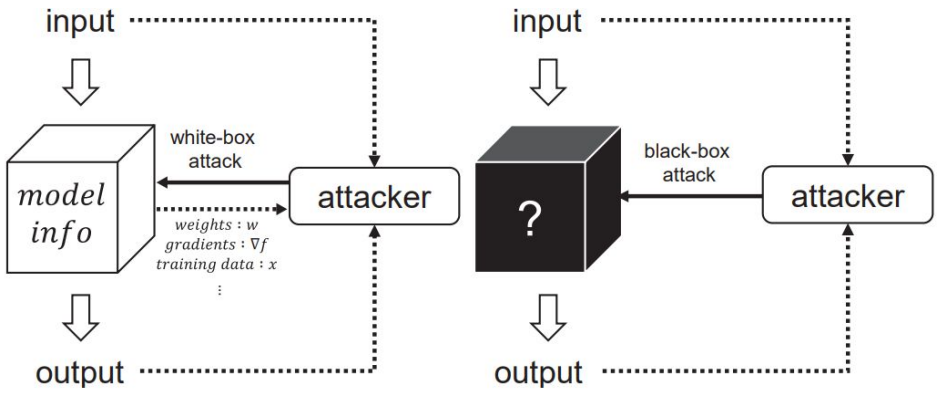
L'article[37] présente quelques attaques de tecniques Deep Learning, en boîte blanche ou boîte noires. Il propose la classification suivante :

- Les attaques par empoisonnement, (*poisoning attack*), des données malveillantes sont introduites dans les données d'apprentissage

- Les attaques par évasion (*evasion attack*), des données malveillantes sont présentées aux entrées du réseau

-Les attaques par inversion (*inversion attack*) visent à reconstruire les données d'apprentissage à partir du modèle

-Le vol des données d'apprentissage (data set) ou du modèle

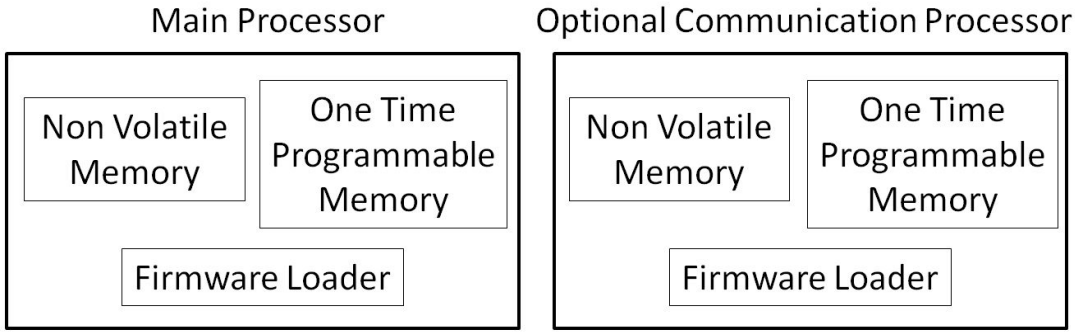


[36] Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P., "Gradient-based learning applied to document recognition", Proceedings of the IEEE. 86 (11): 2278–2324, 1998

Systèmes

Systèmes embarqués

Un dispositif ("*device*" IoT) est typiquement construit à partir d'un processeur principal auquel sont reliés des capteurs et des actuateurs. Ce dernier peut réaliser également des fonctions de communication (*Bluetooth*, *Wi-Fi*); dans le cas contraire un processeur de communication SoC ("*System on Chip*") est nécessaire.



Les objets connectés sont distribués par des canaux grand public ce qui rend possible la modification des logiciels embarqués ("*firmware*") par des attaques qualifiées de "*supply chain attacks*[38]". L'intégrité des micro-logiciels et leur mise à jour ("*update*") est un pré-requis critique pour le déploiement de l'internet des choses. A l'IETF le groupe de travail SUIT (*Software Updates for Internet of Things*) définit des protocoles dédiés à cet usage.

Un processeur d'objet est équipé de mémoires non volatiles de type ROM (non effaçable) ou FLASH (effaçable). Plus précisément pour les mémoires ROM on utilise des technologies de type OTP (*One Time Programmable*) basées sur des fusibles à usage unique. La programmation de mémoire FLASH implique la mise en d'œuvre d'un programmeur et d'un protocole particulier, tel que JTAG (*Joint Test Action Group*), protocole parallèle (utilisant un ensemble de signaux dédiés), ou SPI (*Serial Peripheral Interface*). SPI est un protocole très répandu basé sur deux canaux séries et quatre signaux: SCLK (serial clock), MOSI (master out slave in), MISO (master in slave out), et SS (slave select). En phase de production les logiciels sont chargés dans les mémoires non volatiles. Afin de permettre des mises à jour logicielles un composant particulier, le *bootloader* constitue le premier élément d'une chaîne de chargement. Par exemple le bootloader comporte un pilote USB et un interpréteur de

[37] Ho Bae, Jaehee Jang, Dahuin Jung, Hyemi Jang, Heonseok Ha, Hyungyu Lee, and Sungroh Yoon, "Security and Privacy Issues in Deep Learning", 2020

[38] Voir par exemple https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/supply-chain-malware

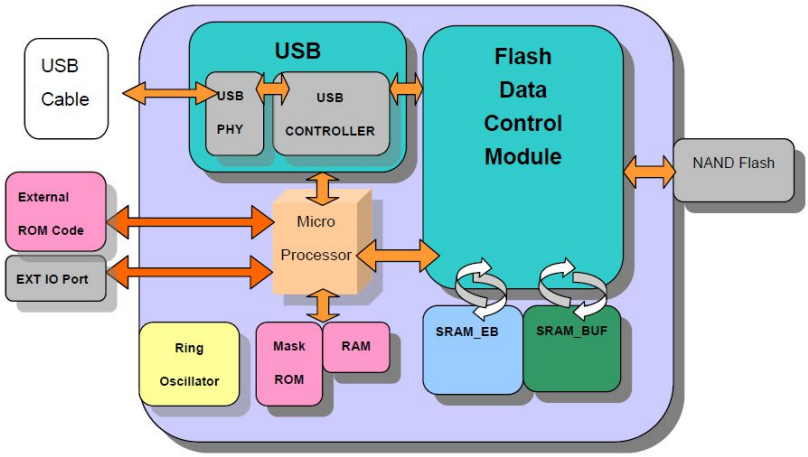
commande; il peut également intégrer des procédures de sécurité permettant la vérification de signature avec une clé publique ou le déchiffrement de données avec une clé symétrique (par exemple selon l'algorithme AES-CCM).

Selon les capacités du processeur le bootloader est logé en ROM ou en FLASH.

Voici quelques cas pratiques qui illustrent les attaques par défaut d'intégrité.

FLASH Disk

Les disques FLASH ("clés USB") sont basés sur des contrôleurs dédiés (*FLASH Controller*) dont les spécifications ne sont généralement pas publiques. Par exemple le contrôleur PS2251-33 de *Phison Electronics Corporation*, illustré par la figure ci-dessous comporte une ROM interne et ROM externe qui stockent un bootloader.



Des sites dédiés (flashboot.ru, www.usbdev.ru...) gèrent des bases de données relatives aux micro-logiciels des clés USB et fournissent les outils nécessaires à leur chargement. Le contenu des mémoires peut être protégé par un mot de passe utilisateur[39] selon une procédure gérée par le micrologiciel. En 2014 une équipe de chercheurs[40] à démontré à la conférence Black Hat des modifications de micrologiciel réalisant l'émulation de clavier (attaque *Key Loger*) ou de cartes réseau (attaque *Proxy*). Les sites[41] [42] détaillent des mises en œuvre telles attaques.

Dongle Bluetooth

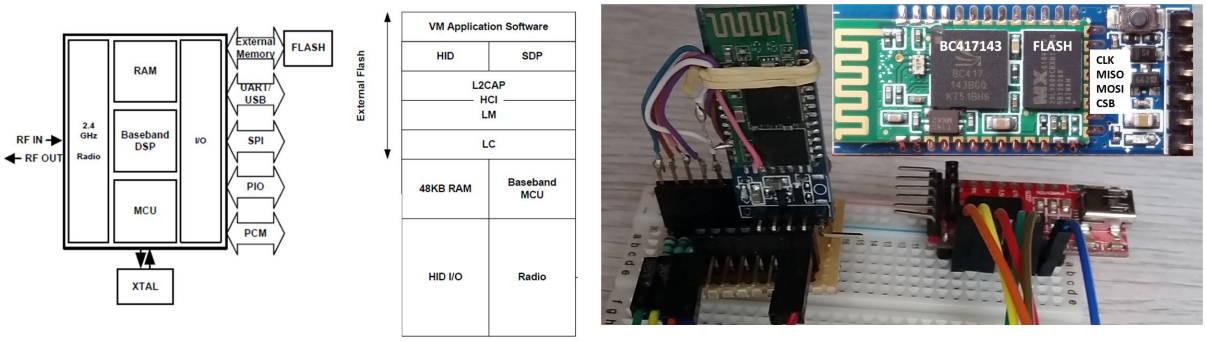
La famille de jetons bluetooth HC05/HC06 est basée sur le SoC *BlueCore™-External* (CSR BC417143), produit par la société *Cambridge Silicon Radio* rachetée en 2015 par Qualcomm. Le SoC est associé à une mémoire FLASH externe, dont le contenu peut lu ou chargé avec le logiciel Blue Flash Software version 2.62, qui nécessite un adaptateur USB dédié. La documentation du SoC indique qu'une interface SPI permet de programmer le premier Megabit de mémoire FLASH, et

2018, Security analysis of USB drive, Master's thesis report,
., KriBler, S., Lell, J. 2014. "BadUSB - On Accessories that Turn Evil", USA

[41] https://github.com/brandonlw/Psychson

[42] https://www.pentestingshop.com/recover-a-usb-stick

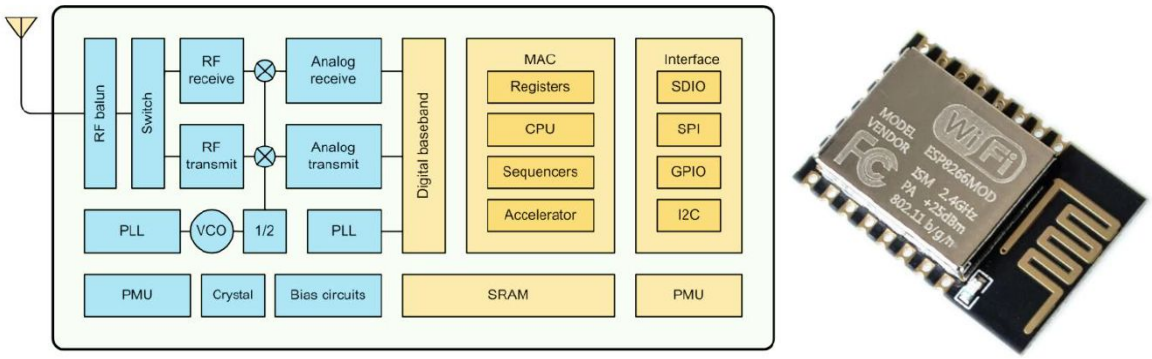
suggère de mettre en œuvre un bootloader. Ce dernier assure par la suite le chargement de l'espace mémoire. La référence[43] commente un reverse engineering du logiciel Blue Flash, et propose une bibliothèque permettant le chargement et la lecture du micrologiciel.



SoC CSR BC417143 (à gauche) et dispositif de chargement du firmware (à droite)

Dongle Wi-Fi

La puce ESP8266 est un SoC Wi-Fi de faible coût, fabriqué par la société *Espressif Systems*. Il est basé sur un microprocesseur RISC 32 bits de la société *Tensilica*, fonctionnant à 80 MHz.



Aucune spécification détaillée n'est disponible publiquement. Néanmoins, certains sites Web fournissent des informations sur les mémoires physiques. Le SoC intègre une ROM de 64 Ko, un ensemble de RAM (environ 80 Ko, y compris le cache des instructions), 80 Ko de mémoire DRAM pour les données utilisateur et une mémoire FLASH externe (jusqu'à 16 Mo). Certains ESP8266 supportent un boot sécurisé, c'est-à-dire que le micrologiciel stocké dans la FLASH est chiffré par une clé AES, gravée dans une mémoire OTP.

La ROM contient un bootloader qui réalise à l'aide d'une fonction UART et d'un logiciel dédié (*ESP FLASH Download Tool*) le chargement du micrologiciel.

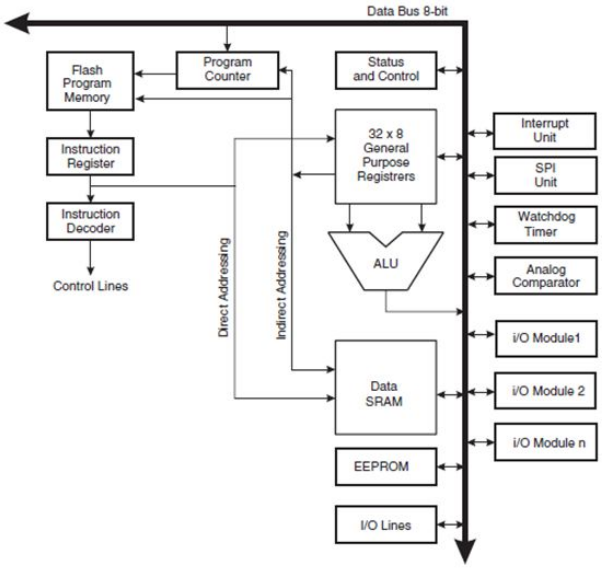
Espressif Systems fournit un kit de développement logiciel sans OS (SDK NONOS) comportant des bibliothèques compilées (notamment une pile TCP/IP) et

[43] Willem, F., 2016, CSR BlueCore USB SPI programmer/debugger, https://github.com/lorf/csr-spi-ftdi

des codes sources ouverts. La plupart des modules ESP8266 sont fabriqués par la société Ai-Thinker sous la marque "ESP-xx".

AVR

AVR est une famille de microcontrôleurs 8 bits RISC commercialisés par la société ATMEL rachetée en 2016 par Microchip Technology. L'architecture AVR a été conçue en 1992 par deux étudiants du *Norwegian Institute of Technology* (NTH), Alf-Egil Bogen et Vegard Wollan. Ces processeurs, en particulier utilisés par l'environnement Arduino, sont équipés de mémoires FLASH, RAM, et EEPROM. La société ATMEL a développé un bootloader associé au protocole DFU (*Device Firmware Upgrade*) et mis en œuvre par le logiciel ATMEL FLIP.



La sécurité des puces AVR est assurée par deux types de registres: les verrous (*locks*) et les fusibles (*fuses*). Les verrous sont réinitialisés grâce aux commandes envoyées via des interfaces de programmation telles que SPI. Ils gèrent la politique d'accès à la mémoire FLASH (opérations de lecture et d'écriture), la politique de sécurité des fusibles (lecture et écriture), et l'utilisation optionnelle d'un bootloader. Les fusibles ne sont pas effacés par la commande de réinitialisation (reset) SPI; ils sont modifiés par des commandes dédiées. Ils contrôlent certaines fonctionnalités de programmation telles que l'utilisation de la broche RESET ou du protocole SPI, ainsi que d'autres paramètres physiques relatifs aux horloges et tensions logiques.

En l'absence de ROM, il n'est pas possible de désactiver définitivement les opérations d'écriture FLASH. En d'autres termes, il est toujours possible d'effacer et de télécharger du code dans la mémoire FLASH. Des outils ouverts telles que AvrDude (AVR Downloader/UploaDEr) ou gratuits (Atmel Studio) sont dédiés à la programmation des processeurs AVR.

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 inter-entreprises.

- 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, contient le noyau, les librairies natives, l'environnement de la Dalvik Virtual Machine (DVM), le framework Java (JNI), et les applications.

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

La sécurité android repose sur *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.

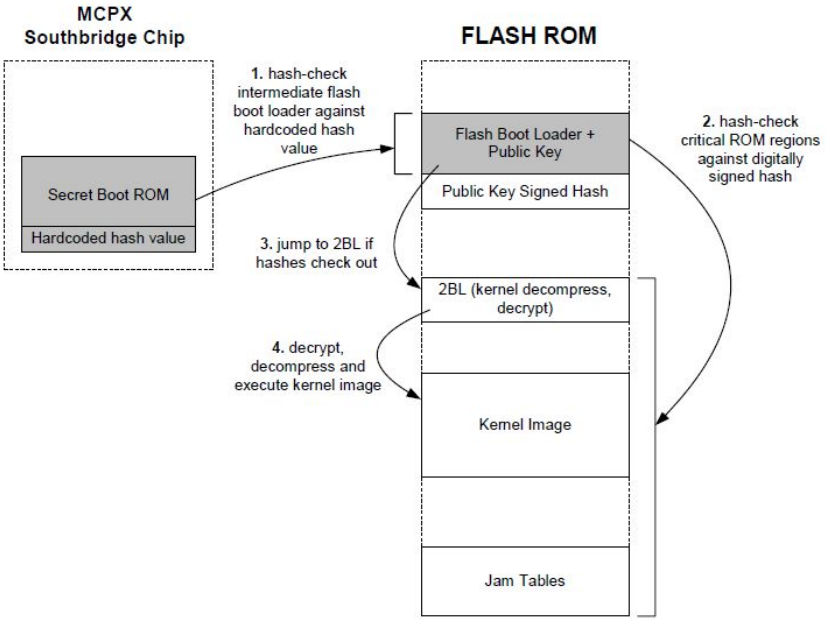
L'intégration du TPM dans le système d'exploitation *Windows* s'applique aux trois points suivants:

- 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*^{TM}) 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*).

Cas d'usage: la sécurité des consoles de jeu XBOX (2002)



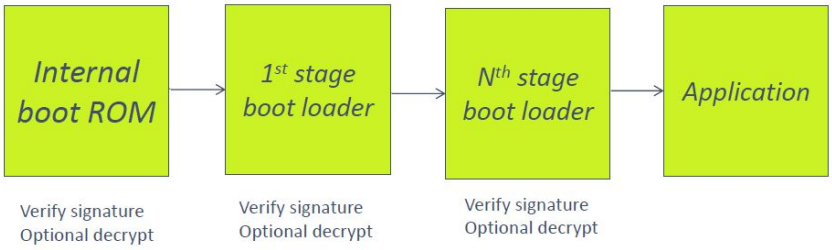
Secure Boot, XBOX 1.1

Andrew Huang a publié[44] en 2002 le reverse engineering de la sécurité de la console de jeu XBOX 1.0. Une ROM de faible capacité (512 octets) loge un code RC4, un interpréteur de commande, et réalise le déchiffrement et le calcul du hash d'un code de boot de 2ième niveau stocké dans une mémoire flash. Il est important de noter d'un part que lorsque la valeur d'un hash de référence est contenue dans la ROM, le contenu du boot de 2ième niveau ne peut être modifié, et d'autre part que le stockage de la clé cryptographique (RC4) est un problème critique.

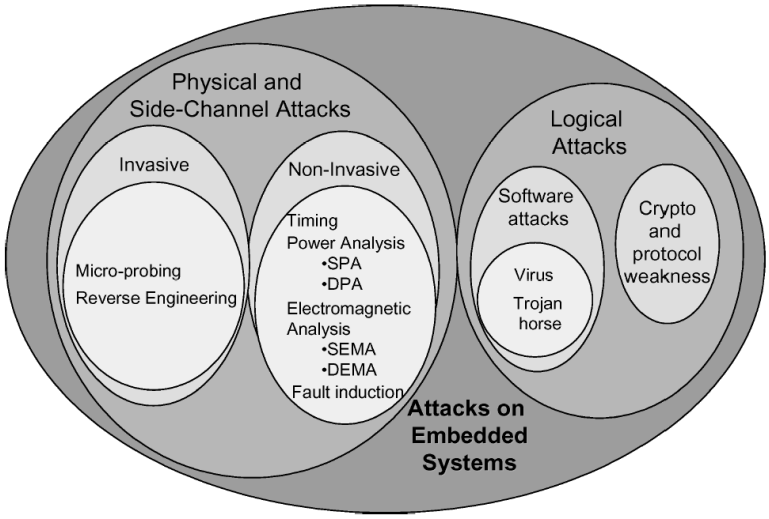
En octobre 2002 Andy Green a publié une analyse de la sécurité de la console de jeu XBOX 1.1. La ROM stocke et vérifie le hash d'un code stocké en mémoire flash. Ce dernier loge une clé publique et les ressources logicielles nécessaires pour la vérification de signature RSA; il déchiffre et exécute un code de boot de 2ième niveau , dont il vérifie l'intégrité.

De manière générale un *secure boot* consiste en une succession d'exécution de blocs logiciels, optionnellement chiffrés, après vérification de leur intégrité.

[44] Hacking the Xbox An Introduction to Reverse Engineering HACKING THE XBOX Andrew "bunnie" Huang, 2002



Exemple de classification des attaques pour les systèmes embarqués.

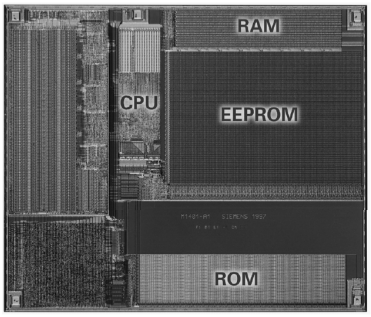


Examples of attack threats faced by embedded systems.

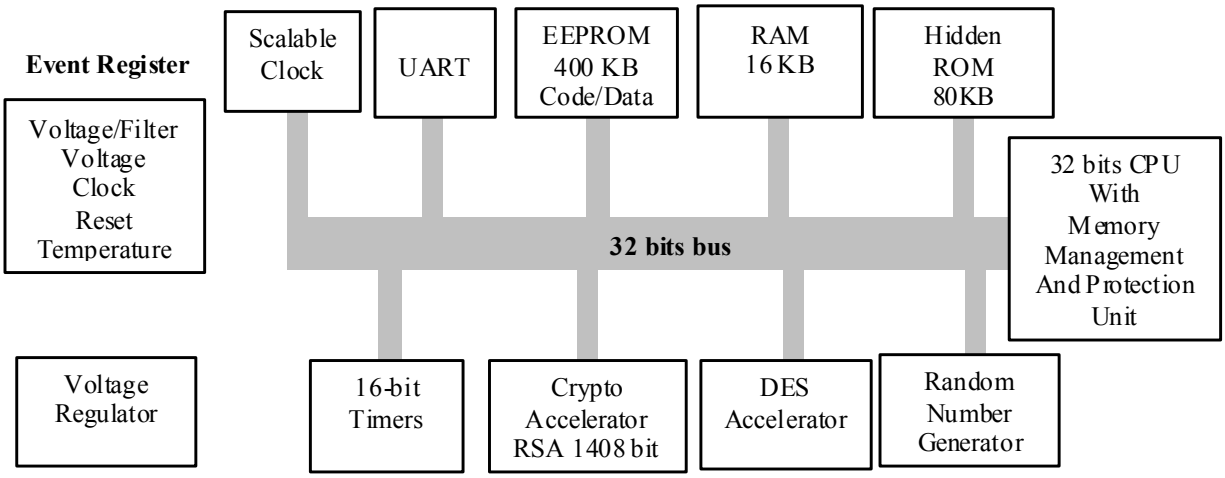
S. Ravi et al, 2004[45]

[45] Ravi, S., Raghunathan, A., Kocher, P. and Hattangady, S., 2004. Security in embedded systems: Design challenges. ACM Transactions on Embedded Computing Systems (TECS), 3(3), pp.461-491.

Secure Elements



Un *secure element* est un microcontrôleur sécurisé[46] [47] (de surface inférieure à 5x5 mm^2), muni d'une interface de communication (série, USB,…) qui résiste à des attaques physiques et logiques. Bien que des composants de 16 ou 32 bits existent, le CPU le plus fréquemment utilisé est basé sur une architecture 8 bits (par exemple le 6502, microprocesseur de l'Apple II). La taille des mémoires embarquées est de l'ordre d'une dizaine de KB pour la RAM, 100KB pour la FLASH qui contient le code des applications, et quelques centaines de 100Ko pour la ROM qui stocke un système d'exploitation.



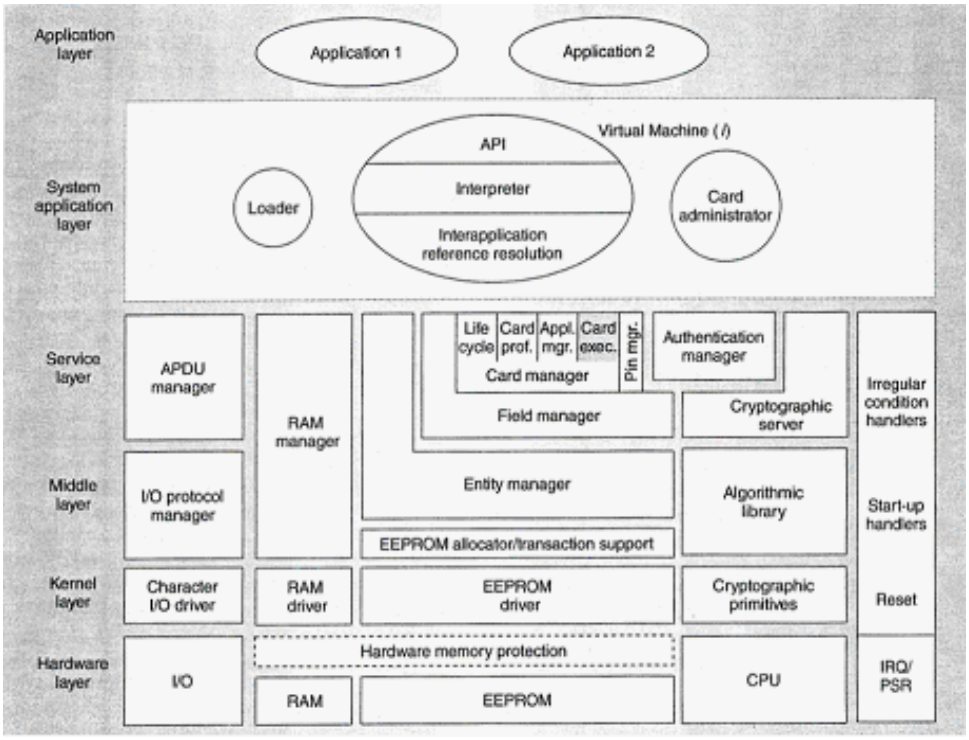
Les attaques physiques

- La pose de microsondes à la surface du circuit. L'attaquant désire obtenir les secrets stockés dans la mémoire non volatile, par exemple en l'isolant la mémoire du reste de la puce sécurisé, et en produisant les signaux électriques nécessaires à sa lecture;
- Le reverse engineering, reconstruction du *layout* de la puce, visualisation du code ROM, et plus généralement identification des contres mesures;
- L'injection de faute. Grâce à des interactions physiques (coupure d'alimentation, *glitches* de tension, injection laser, etc..) on perturbe le fonctionnement normal du microcontrôleur, dans le but de produire des erreurs permettant de déduire la clé d'un algorithme cryptographique ou de contourner certaines procédures logicielles (code PIN,…).

[46] Smart Cards - The developer's Kit. Timothy M. Jurgensen, Scott B. Guthery. Editions Prentice Hall

[47] Smart Card handbook. W. ERankl , W. Effing. Editions Willey

- Les contre mesures physiques intègrent le chiffrement du bus système, le chiffrement des mémoires, la détection de paramètres de fonctionnement (température, alimentation, fréquence) anormaux, la perte d'intégrité physique (treilli métallique actif), l'absence de rayonnement radio (plan de masse métallique), un design désordonné (fuzzing design) pour masquer les structures fonctionnelles (mémoires,…).



Structure d'un système d'exploitation de Secure Element[48]

Attaque de Bellcore

Un exemple classique d'attaque par injection est l'attaque dite de Bellcore[49]. De nombreuses implémentations utilisent le théorème du résidu chinois (Chinese Reminder Theorem, CRT) pour le calcul de RSA. Soit n un produit de deux nombres premiers n=p.q, la procédure RSA s'écrit: $x^s \bmod n$, x étant une valeur à chiffrer et s l'exposant de la clé privée.

Le calcul CRT s'obtient par les relations suivantes

$E1= x^s \bmod p$, $E2=x^s \bmod q$

$y = a.E1 + b.E2 \bmod pq$, $a=1 \bmod p$, $a=0 \bmod q$, $b=1 \bmod q$, $b=0 \bmod p$

On réalise un premier calcul sans erreur, on obtient $y = a.E1 + b.E2 \bmod pq$

Un deuxième calcul produit une erreur, on obtient $y' = a.E1' + b.E2 \bmod pq$

$y-y' = a.(E1-E1')$

Si $E1-E1'$ n'est pas divisible par p, $pgcd(y-y', n) = q$.

[48] MASSC: a generic architecture for multi application smart cards. IEEE Micro 19(5): 52-61 (1999)
[49] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the Importance of Checking Computations", Eurocrypt 96.

L'attaque a été améliorée par l'article[50] qui ne requiert qu'une seule signature erronée.

Les attaques logiques

- *Les attaques temporelles* (moyenne, écart type). Certaines implémentations d'algorithmes présentent des temps de calcul différents en fonction des valeurs d'entrée et de la clé utilisée.
- *Les attaques par corrélation statistique,* telles que *Simple Power Attack* (SPA) ou *Differential Power Analysis* (DPA). Un processeur réalise un algorithme à l'aide d'une suite d'opérations nécessairement différentes en fonction de la clé. Ainsi un algorithme utilisant une clé parmi 2^p possibles, traite au moins p informations différentes pour une clé particulière. Il produit donc des signaux électriques (par exemple puissance consommée) ou radioélectriques qui sont corrélés à la clé opérationnelle.

Beaucoup de secure elements intègrent[51] une machine virtuelle java (JVM) qui exécute des applications écrites en javacard. Les applications sont chargées via un canal sécurisé (*Secure Channel*), normalisé par les standards Global Platform.

Trusted Execution Environment

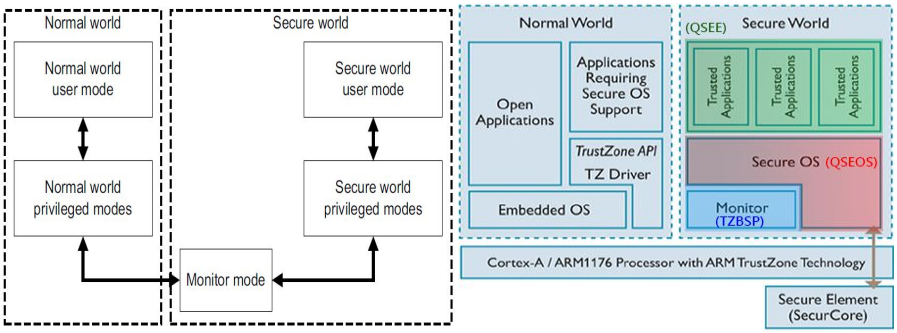
Un processeur TEE, par exemple TrustZone[52] de la société ARM, réalise la virtualisation d'un processeur permettant la création de deux espaces de traitement de l'information le "*Normal Word*" et le "*Secure World*". Chacun de ces modes de fonctionnement possède un banc de registres séparé et des mécanismes d'interruption différents. Cependant le CPU et les mémoires internes (ROM, SRAM) sont communs aux deux mondes. Une troisième entité, le Monitor gère les changements de contexte entre le "Normal Word" -NW- et le "Secure World" -SW- à l'aide d'instructions spécifiques (Secure Monitor Call, SMC); elle se comporte de fait comme un hyperviseur qui réalise grâce à la technique de virtualisation, une isolation des espaces dits normaux ou sécurisés.

---

50 A. K. Lenstra, "Memo on RSA Signature Generation in the Presence of Faults", Manuscript, Sept. 28, 1996. Available from Author at arjen.lenstra@citicorp.com

51 Java Card™ Technology for Smart Cards. Zhiqun Chen. Editions Addison Wesley

52 ARM Security Technology, Building a Secure System using TrustZone® Technology", 2009, http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf



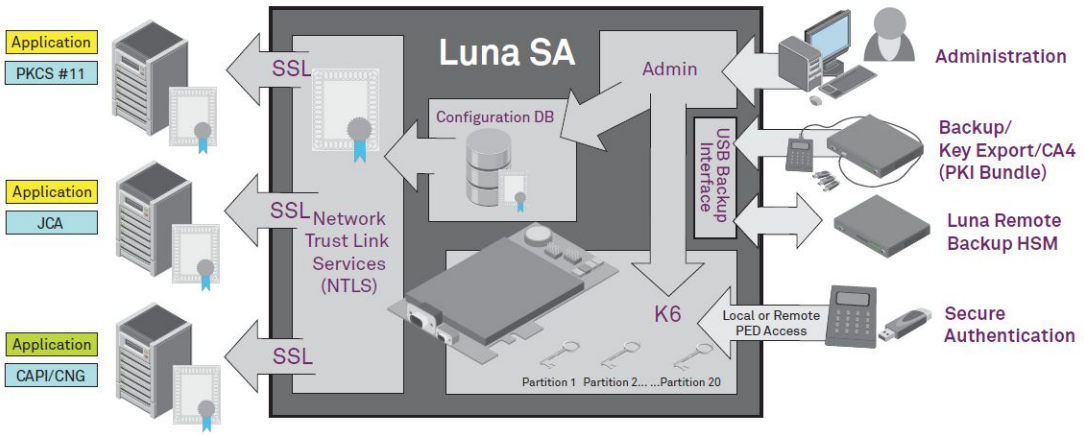
Les tailles mémoire internes sont de l'ordre de 10 Ko pour la ROM et 128 Ko pour la SRAM. Ces capacités limitées expliquent la compétence native des industriels de la carte à puce pour la conception de systèmes d'exploitation sécurisés compatibles avec ces modestes ressources.

D'un point de vue physique, et contrairement aux Secure Elements, le processeur n'implémente pas de contremesures matérielles. ("*Hardware techniques and processes used for smartcards are impractical for standard Soc designs*…"). La technologie SoC permet cependant d'intégrer des blocs cryptographiques démunis de protections hardware.

Les mémoires externes non volatiles (ROM, FLASH) ou volatiles (DRAM…) sont partagées par les deux mondes. Une entité MMU (*Memory Management Unit*) réalise les partitions nécessaires à leur virtualisation; des protections cryptographiques (chiffrement et intégrité) sont nécessaires pour la sécurité des informations stockées par le "Secure World". Le MMU assure également les partitions mémoires internes au SoC.

La référence[53] détaille une injection de code arbitraire dans l'environnement de confiance QSEE, exploitant une faille de type *buffer overflow* détectée dans le *trustlet Widevine*.

Hardware Secure Module (HSM)



Un Hardware Secure Module est un système informatique "durci", qui résiste à des attaques physiques, avec des assurances de sécurité de EAL1 à EAL4, selon une

[53] https://bits-please.blogspot.fr/2016/05/qsee-privilege-escalation-vulnerability.html

échelle Critère Commun (CC) qui comporte 7 niveaux. Sa fonction est d'exécuter des algorithmes cryptographiques symétriques ou asymétriques. Il est conçu par exemple pour effacer les clés cryptographiques lors d'une intrusion physique.

Le standard FIPS 140-2 définit quatre niveaux de sécurité pour les HSM

Le niveau 1 utilise un algorithme cryptographique normalisé, exécuté sur un système d'exploitation standard. Une carte de chiffrement peut être mise en œuvre.

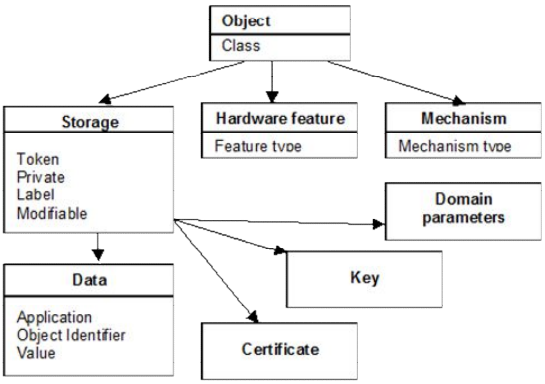
Le niveau 2 utilise un boitier physique fermé et sécurisé (notion de *tamper resistance*). Il met en œuvre un contrôle d'accès au module cryptographique, basé sur les rôles (*Role-Based Access Control* - RBAC); il est exécuté sur un système informatique classique dont le système d'exploitation répond aux exigences fonctionnelles spécifiées dans les profils de protection (PP) critères communs (CC). Il est associé à un profile de protection EAL2.

Le niveau 3 implémente la détection des intrusions physiques, impliquant l'effacement des secrets. Il met en œuvre une authentification des rôles basée sur l'identité. Des interfaces d'entrée/sortie dédiées doivent être disponibles pour l'administration. Il est associé à un profile de protection EAL3.

Le niveau 4 réalise la détection d'intrusion logique et physique renforcée, par exemple il identifie les paramètres fonctionnels (température, alimentation, horloge …) anormaux, et implémente des contremesures. Il est associé à un profil de protection EAL4.

PKCS11

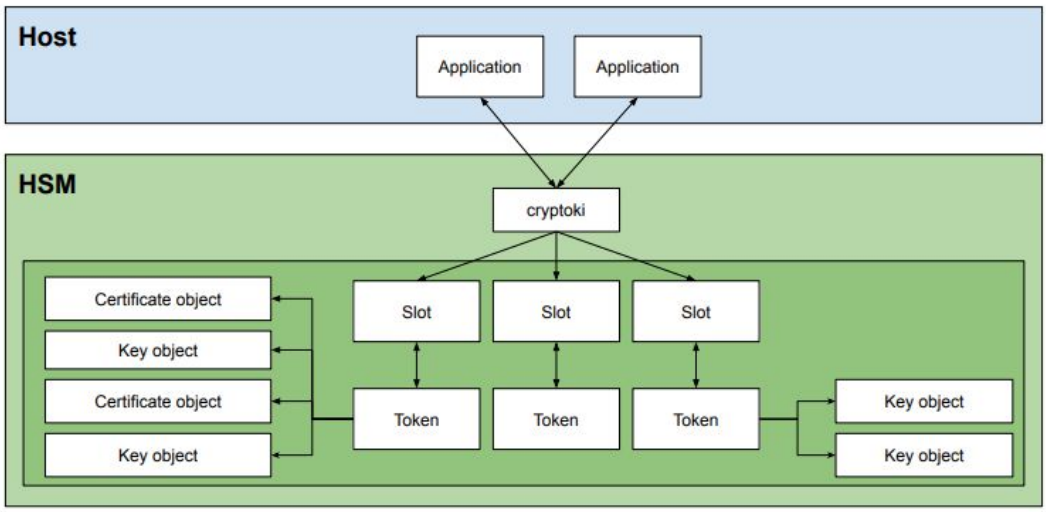
Le standard PKCS11, issue de l'entreprise RSA Laboratories, est depuis 2013 mis à jour par l'organisation *OASIS PKCS11 Technical Committee*. PKCS11 est une API cryptographique, communément dénommée *Cryptographic Token Interface* (cryptoki)



Cryptoki organise les ressources cryptographiques selon une hiérarchie de slot et de token. A l'origine le slot représentait un lecteur de carte à puce et le token une carte physique. Les sessions sont ouvertes avec un couple (slot,token), et une authentification optionnelle basée sur un PIN code ou sur un mécanisme de défi-réponse.

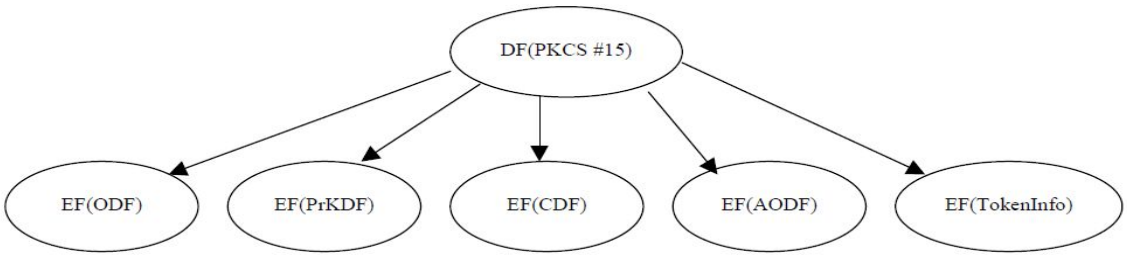
Les token hébergent des objets selon trois classes principales: les clés les certificats et les données. Ces objets possèdent des paramètres de domaine cryptographique, et des attributs relatifs à leur sécurité (SENSITIVE, valeur secrète, EXTRACTABLE, PRIVATE authentification nécessaire). Les *mécanismes* définissent un mode opératoire cryptographique (chiffrement, déchiffrement, signature…). Deux types d'utilisateurs sont définis (rôle), l'utilisateur ordinaire (*user*) et les officiers de sécurité (*Security Officier*, SO),

utilisant des codes PINs différents. Le SO initialise un token et les codes PINs utilisateurs. Il ne peut cependant pas accéder aux objets privés.



PKCS15

Le standard PKCS15 décrit l'organisation logique d'un jeton (carte à puce par exemple) sous forme d'un ensemble de répertoires et de fichiers (ODF object directory file, PrKD private key directory file, CDF certificate directory file, AODF authentication object directory file, DODF data object directory file).



HSM hacking

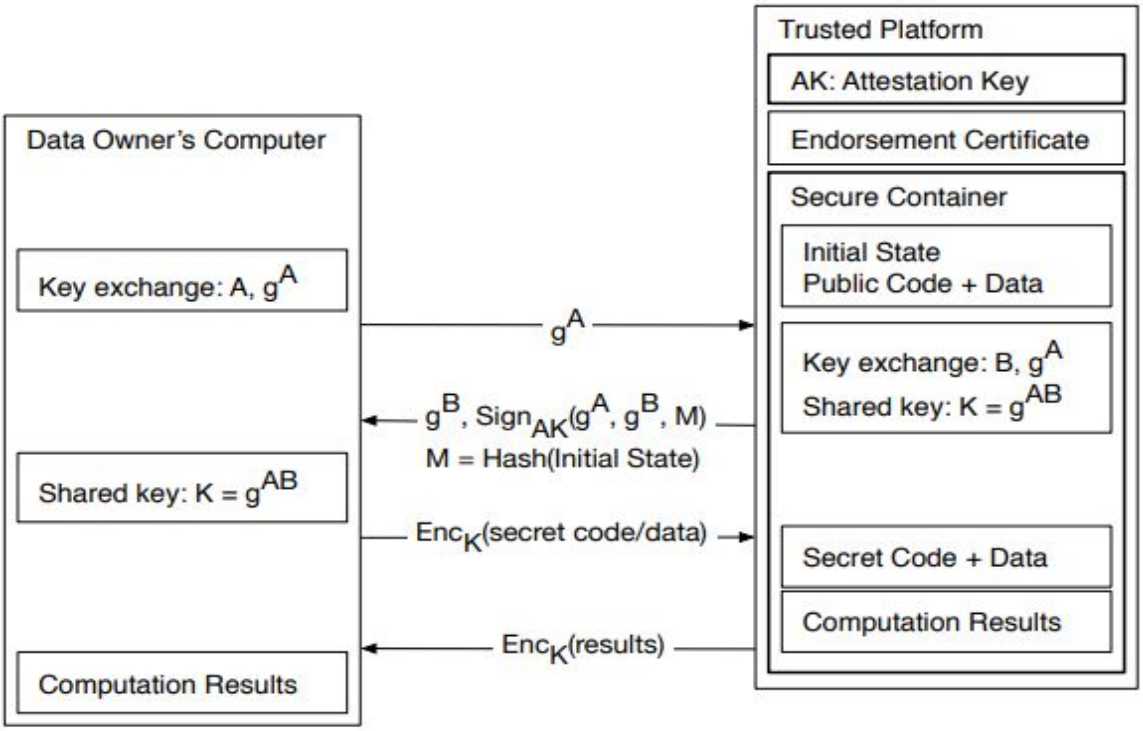
D'un point de vue logique un HSM est un ensemble de slot et de token. L'article[54] décrit le reverse-engineering .d'un HSM certifié FIPS 140-2 niveau 3. Il se présente sous forme d'un système UNIX exécutant une application pkcs11, dans une carte PCI équipé d'un processeur powerPC et recouverte de résine époxy. Les données sensibles sont chiffrées au moyen d'une clé maitre stockée dans une mémoire SRAM alimentée par une batterie. En cas d'intrusion la mémoire est effacée, et par conséquent il devient impossible de déchiffrer les données protégées. La communication avec le HSM utilise des mémoires DRAM partagées. La présence d'un *buffer overflow* dans le parseur de commande, permet d'injecter du code malveillant réalisant la lecture de l'ensemble des informations stockées dans le HSM. De surcroit l'attaque s'appuie sur des commandes non authentifiées.

[54] Jean-Baptiste Bédrune et Gabriel Campana, "Everybody be cool, this is a robbery!", SSTIC 2019

Intel SGX

La technologie SGX[55] [56] [57] [58] est une exécution isolée dans un processeur INTEL multi-cœurs, selon la notion d'enclave. Elle est conçue pour résister aux malwares présents au niveau du système d'exploitation. Un logiciel (ISV) crée une enclave à partir d'instructions dédiées SGX. L'identité de l'enclave ("*Enclave Identity*") est l'empreinte mémoire (hash) du logiciel et d'autres données. Le "*Sealing Identity*" est un certificat émis par le "*Sealing Authority*" pour certifier "*Enclave Identity*", ainsi que d'autres informations, par exemple un identifiant de produit.

L'attestation logicielle prouve à un ordinateur distant qu'il communique avec un conteneur sécurisé (son image mémoire est chiffrée,) spécifique hébergé par une plate-forme de confiance. La preuve est une signature d'attestation produite par la clé d'attestation secrète de la plate-forme. La signature couvre l'état initial du conteneur, un nonce de défi produit par l'ordinateur distant et un message produit par le conteneur.



En phase de production deux secrets hardware sont générés :

- Le *root provisioning key*

- Le *root seal key* , qui n'est pas connu d'INTEL

Le *root seal key* est utilisé pour calculer des *seal keys* pour les enclaves

[55] Victor Costan and Srinivas Devadas, Intel SGX Explained, 2016

[56] JP Aumasson, Luis Merino, SGX Secure Enclaves in Practice Security and Crypto Review, Blackhat 2016

[57] https://software.intel.com/content/www/us/en/develop/articles/innovative-technology-for-cpu-based-attestation-and-sealing.html

[58] https://sgx101.gitbook.io/sgx101/sgx-bootstrap/overview

L'Attestation prouve l'intégrité d'une enclave

- Une enclave particulière *Quoting Enclave* (QE) génère une clé de signature asymétrique EPID.

- QE mesure l'intégrité d'une enclave et délivre une attestation de vérification.

Notion de Sealed storage

- Après la création d'une enclave les données internes de l'enclave sont protégées au runtime par une clé *Seal Key*

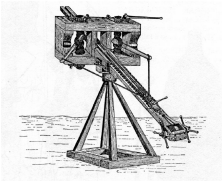
- Des secrets peuvent être crées par l'enclave et stockés à l'extérieure de l'enclave selon un mécanisme dit *Encrypted blob*.

Eléments de sécurité

Quelques paradigmes de sécurité

Classification des types d'attaques

On peut classer[59] les attaquants d'un système en trois catégories:



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

- **Classe II -** (*knowledgeable insiders*). L'attaque individuelle ou par de petites communautés (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'antibiotique 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[60] (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.

---

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

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

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

Une Méthodologie de Cyber Attaque: Empreinte, Collecte, Inventaire

Dans les différentes éditions du best seller[61] "*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.

Au sujet des normes Critères Communs (CC)

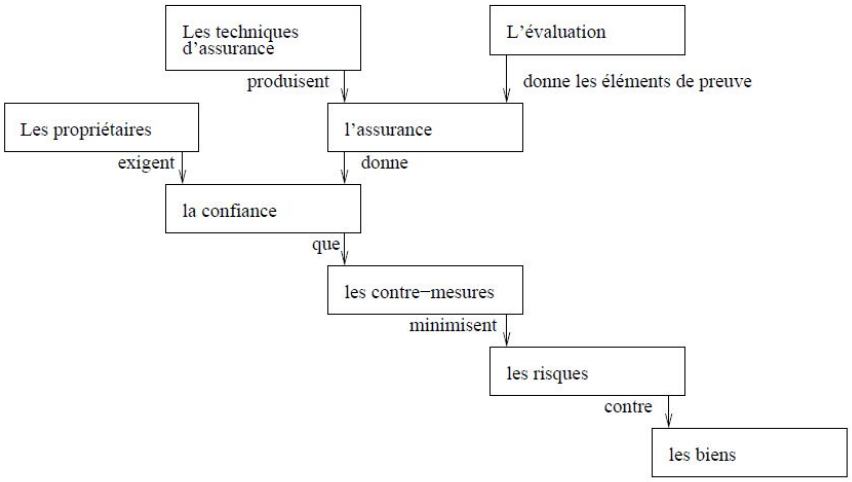
Les normes ISO 15408 dénommées *Critères Communs* visent à qualifier la sécurité de produits tels que microcontrôleurs sécurisés, cartes à puce ou firewalls. Elles sont organisées en trois parties, introduction (partie 1), exigences fonctionnelles de sécurité (partie 2) et exigences d'assurances de sécurité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

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



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.

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.

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 chiffrement soit en mode flux (octet par octet, comme par exemple dans RC4, AES-CTR) 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), SHA2, SHA3 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 combinés 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)

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 (étatique, 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, mot de passe, ….

- Ce que je possède, carte à puce, jeton USB…

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 (CNIE).

- **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, FIDO, Titan Key…)

Google Titan Key

Taxonomie des méthodes d'authentification

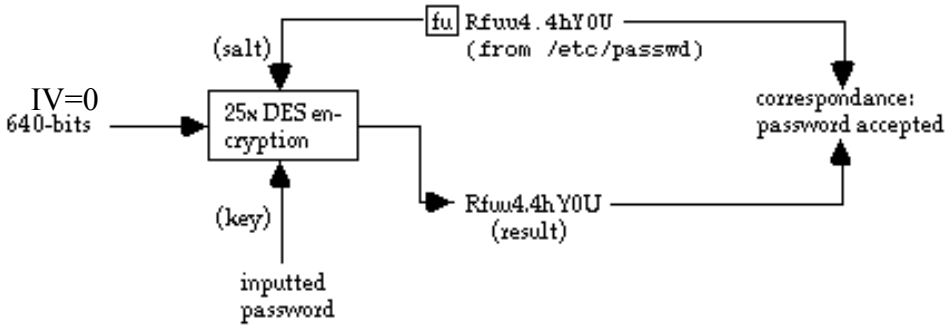
Nous classons les méthodes d'authentification en trois catégories, symétriques (secrets partagés), asymétriques (basées sur RSA ou ECC en règle générale) et tunnels.

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

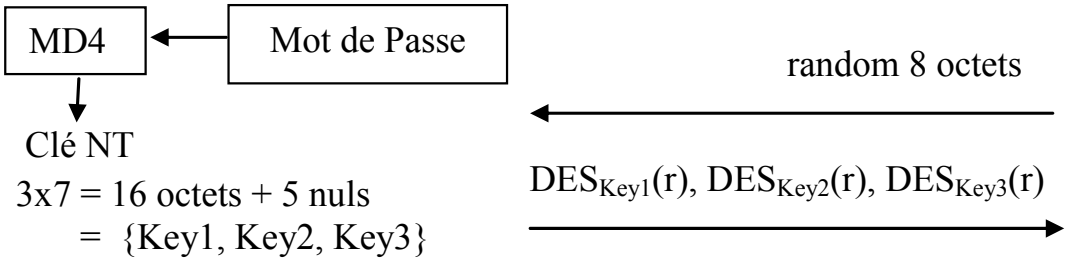
Nous divisons les méthodes symétriques en trois classes

- **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.



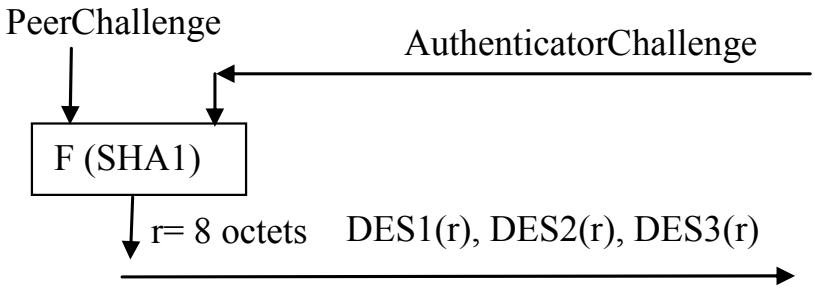
Dans l'univers Microsoft la sécurité d'un ordinateur personnel est fortement corrélée au mot de passe de son utilisateur. Ce dernier n'est jamais stocké en clair dans la mémoire de la machine. A partir d'un mot de passe on calcule une empreinte MD4 de 16 octets, mémorisée par le système hôte. Cette valeur, 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).



MSCHAPv1

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.



MSCHAPv2

Sous Windows le SAM (*Security Account Manager*) est la base de données 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[62] 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, 3G, 4G, 5G une base de données 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.

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

Les tunnels

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

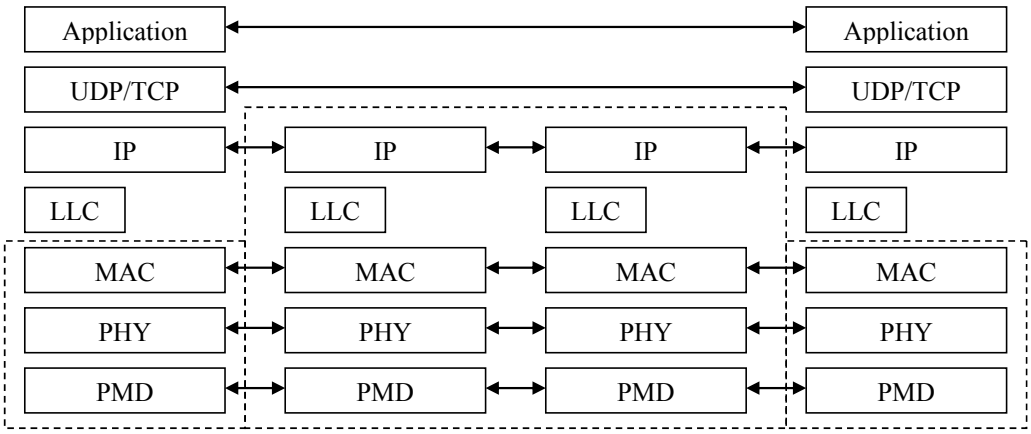
Les tunnels, s'appuyant fréquemment sur la technologie SSL/TLS, protègent un dialogue d'authentification grâce au chiffrement des données échangées. Il existe aujourd'hui de multiples standards devenus nécessaires, en raison des nombreux logiciels disponibles sur le WEB qui cassent les protocoles à base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

Réseaux

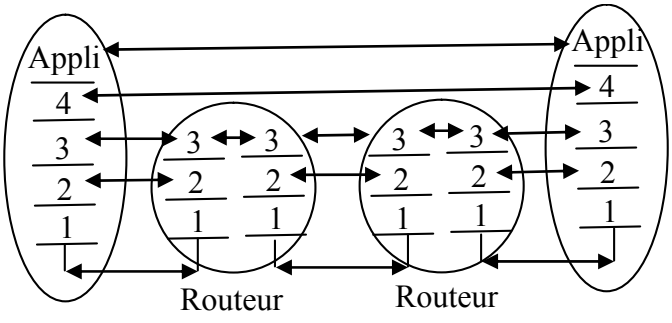
Sécurité des Réseaux

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 classons 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

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

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

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

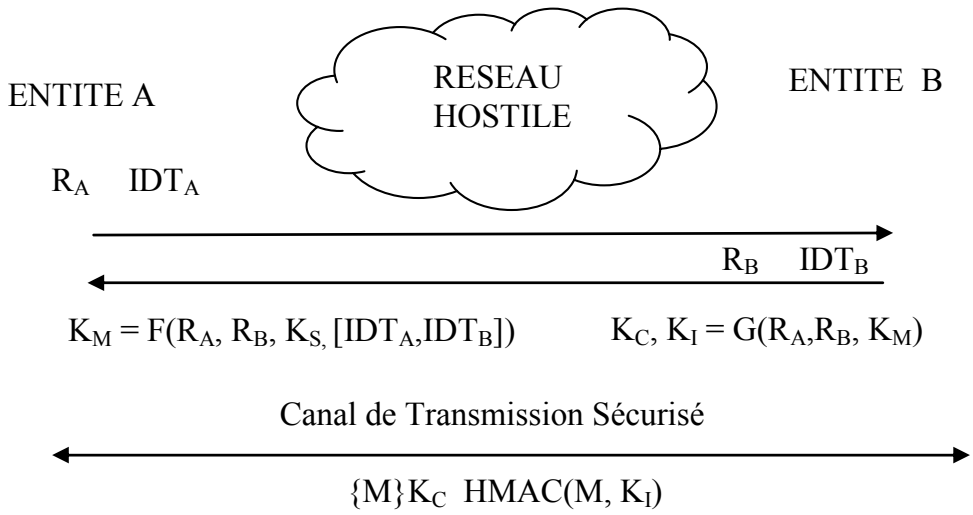
Insertion d'une couche de sécurité additive assurant la protection d'application telles que navigateurs WEB ou messageries électroniques. Par exemple le protocole SSL/TLS basé sur la cryptographie asymétrique réalise cette fonction. Généralement ce dernier conduit une simple authentification entre serveur 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

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 informatique, 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 (*keying*).

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 mod 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.

Sécurité du canal

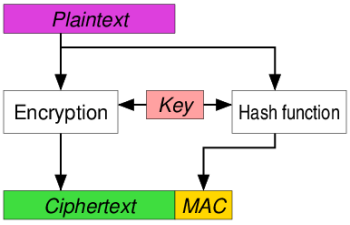
Un schéma cryptographique assurant à la fois intégrité et confidentialité garantit la sécurité contre les attaques à chiffrés choisis IND-CCA (*Chosen-Ciphertext Attack*). Deux propriétés sont nécessaires :

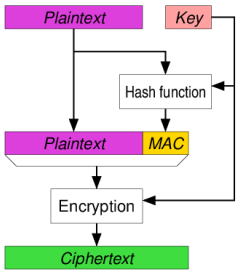
- IND-CPA= Chosen Plaintext Attack

- INT-CTXT= Integrity of CipherTeXT

En d'autres termes il faut chiffrer les données et vérifier leur intégrité avec un MAC (Message Authentification Code). Plusieurs méthodes sont possibles pour ces opérations: quel niveau de sécurité peut on espérer[63] ?.

L'algorithme MAC-And-Encrypt, soit E(M)||MAC(M), n'est pas sûr. Le MAC du message (M) est calculé avant chiffrement, le paquet est la concaténation du MAC et du message chiffré.





L'algorithme MAC-Then-Encrypt, soit E(M||MAC(M)) n'est pas génériquement sûr, mais en pratique on peut construire des schémas sûrs (avec des preuves de sécurité), tels que AES-CCM.

Les versions TLS 1 et 2 utilisent ce schéma, un HMAC est ajouté au message, avec des octets de padding optionnels, puis l'ensemble est chiffré E(M || HMAC(M) || Padding_Bytes). Des attaques telles que *Lucky Thirteen* s'appuient sur le fait que les octets de padding ne sont pas pris en compte dans le calcul HMAC.

Une ancienne version du keystore Android protégeait les clés symétriques par un procédé MAC-Then-Encrypt tel que:

- AES-CBC (MD5(Header || Key) || (Header || Key))

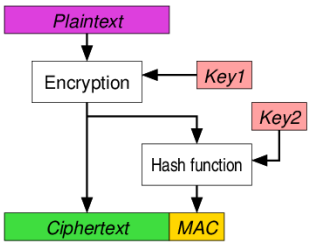
L'article[64] décrit une attaque qui forge des petites clés, basée sur la coïncidence de taille (16 octets) du calcul MD5 et des blocs AES.

L'algorithme Encrypt-Then-MAC, soit E(M)||MAC(E(M)) est sûr, si le mode de chiffrement est sûr, et que le MAC réalise l'intégrité.

---

63 http://www.di.ens.fr/~fouque/mpri/MAC.pdf

64 Mohamed Sabt, Jacques Traoré, "Breaking Into the KeyStore: A Practical Forgery Attack Against Android KeyStore", ESORICS 2016 .



Après une série d'attaques liées à l'usage du mode cryptographique de chiffrement CBC et de l'usage de l'algorithme MAC-Then-Encrypt, le protocole TLS 1.3 a adopté des algorithmes *Authenticated Encryption with Associated Data* (AEAD) tels que AES-CCM, qui utilisent généralement des CMAC (block-cipher-based MAC).

Sécurité des échanges de Diffie Hellman

Considérons un échange de Diffie Hellman (DH) dans Z/pZ*, avec p premier, soit $(g^x)^y$ mod p, g^x est une clé publique (p_k), y une clé privée, et g un générateur d'ordre p-1.

D'après le théorème de Sylow, si l'ordre N d'un groupe se décompose en facteurs premier (qi), $N= \prod qi^{ki}$, il existe des sous groupes d'ordre q^{ki}. La décomposition en facteurs premier de N= p-1 nous montre donc l'existence de sous groupes d'ordre q^{ki}. De surcroit les sous groupes d'un groupe cyclique (*monogène*) sont cycliques.

Dans Z/pZ* il existe des générateurs d'ordre p-1, mais aussi des générateurs pour des sous groupes plus petits dont l'ordre divise p-1. Une attaque possible dans les échanges DH est de proposer une clé publique $p_k= g_d{}^x$ utilisant un générateur dont l'ordre d divise p-1 (d | p-1).

*Comment trouver des générateurs dans Z/pZ**

Il existe φ(p-1) générateurs (φ étant le nombre d'Euler), le nombre maximum de générateurs est (p-1)/2, c'est à dire le nombre d'entiers impairs inférieurs à p. Si p est de la forme $p=1+2^n$, $\varphi(p-1)= \varphi(2^n)= 2^{n-1}=$ (p-1)/2.

Une méthode consiste à trouver des générateurs g_{ki} d'ordre qi^{ki} (il existe $\varphi(qi^{ki}) = (qi-1).q^{ki-1}$ générateurs d'ordre qi^{ki}) puis d'en réaliser le produit. Il existe φ(p-1) générateurs soit $\varphi(p-1)= \prod \varphi(qi^{ki})= \prod (qi-1).qi^{(ki-1)}$.

Safe Prime

Un *Safe Prime* p est de la forme p=2q + 1 avec q premier, q est le *Sophie Germain* prime de p. On en déduit p-1 = 2q , et φ(p-1)= q-1. Il existe un générateur d'ordre 2 (p-1, puisque $(p-1)^2=1$ mod p) et q-1 générateurs d'ordre q.

Si p = 7 mod 8 (voir RFC 7919), p est un diviseur du nombre de Mersenne $Mq = 2^q-1$, on en déduit que 2 est un générateur d'ordre q. Par la suite 2^k avec k ϵ [1, q-1] est un générateur d'ordre q.

Les générateurs d'ordre p-1 sont obtenus par les produits (p-1) 2^k mod p.

Attaque DH et contre mesure

Considérons un groupe d'ordre $N= \prod qi^{ki}$. H un sous groupe d'ordre q premier (q|n), g_q est un générateur de H d'ordre q= qi; par exemple le plus grand qi.

Soit d un diviseur de N, g_d un générateur d'ordre d < q.

Par exemple dans Z/pZ d | (p-1).

$p_k = g_d^x$ x ϵ [1, d] une clé publique malicieuse, $(g_d^x)^d = 1$, il existe d secrets partagés DH = g_d^{xy} y ϵ [1, q]

$p_k = g_q^x$ une clé publique bien formée, c'est à dire un générateur de H d'ordre q, $(g_q^x)^q = 1$ il existe q secrets partagés DH.

Le test $p_k^q = 1$ (implémenté par exemple dans OPENSSL) identifie une clé publique bien formée d'ordre q. Une clé publique malicieuse dont l'ordre d ne divise pas q est détectée par ce test.

Mise à jour des clés cryptographiques

Claude Shannon a défini la notion d'entropie de l'information.

$H = - \sum p(x_i) \log_2 p(x_i)$, soit $\log_2(n)$ pour n symboles équiprobables, pour $n=2^p$ l'entropie est de p bits.

L'entropie conditionnelle de X sachant Y s'écrit:

$H(X,Y) = - \sum p(X=x,Y=y) \log_2 p(X=x,Y=y)$

$H(X,Y) = H(X) + H(Y|X)$, $H(Y|X) = H(X,Y) - H(X)$

Un système cryptographique parfait au sens de Shannon est tel que:

H(M|C) = H(M), M le message en clair et C le message chiffré.

Par exemple une clé K constituée par une suite d'octets aléatoires $R_1,R_2,\ldots,R_i$ réalise un système cryptographique parfait ($C_i = R_i$ exor M_i, $M_i = R_i$ exor C_i), c'est le code de Vernam.

H(C)= p

$p(X=M_i,Y=C_j) = p(X=M_i) \times 1/n = 1/n^2$

D'où H(M,C) = 2p, H(M) = H(C)

En particulier lorsque tous les messages en clair sont équiprobables la taille de la clé doit être au moins aussi grande que la taille des messages en clair. Un algorithme de chiffrement utilisant une clé de taille finie, cette dernière doit être modifiée *de temps à autre*.

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.

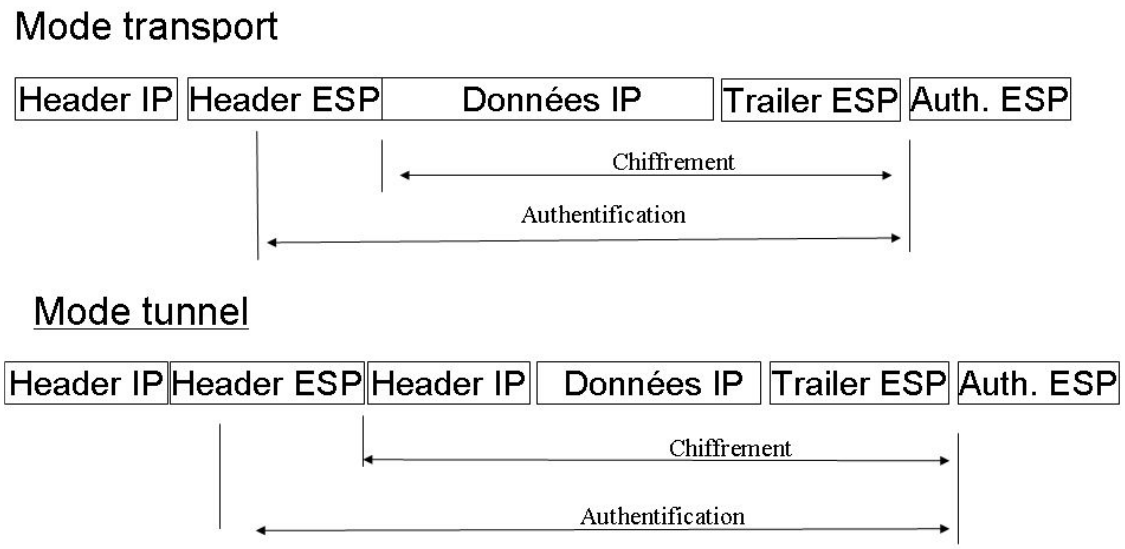
Les solutions sécurisées IP classiques: VPN, TLS, Kerberos, SSH

IPSEC

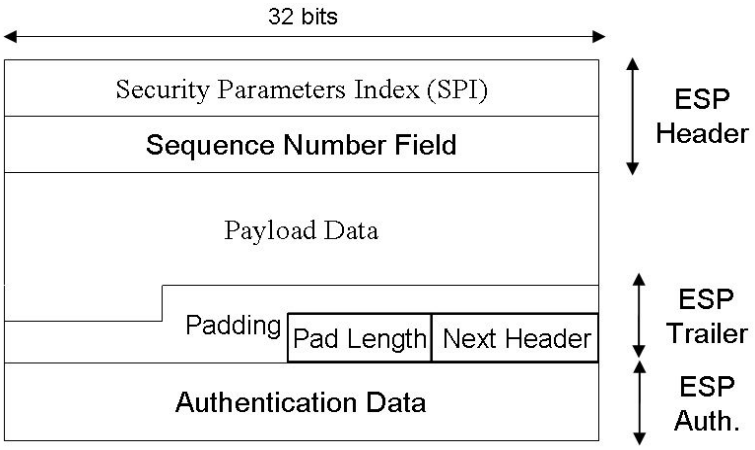
IPSEC assure l'intégrité, l'authentification et la confidentialité des charges transportées par le protocole IP. Les ressources cryptographiques sont déduite d'une **association de sécurité** (SA) basée sur un secret partagé entre les deux entités de la session sécurisée IP. La difficulté de déploiement 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.

```
        Initiator                           Responder
       -----------                         -----------          Echange
        HDR, SA                    -->                         Chiffré
                                   <--     HDR, SA                  |
        HDR, KE, Ni                -->                              |
                                   <--     HDR, KE, Nr              |
      ------------------------------------------------------------  |
        HDR*, IDii, [ CERT, ] SIG_I -->                              |
                                   <--     HDR*, IDir, [ CERT, ] SIG_R
      ------------------------------------------------------------
For pre-shared keys:
              SKEYID = prf(pre-shared-key, Ni_b | Nr_b)

The result of either Main Mode or Aggressive Mode is three groups of
authenticated keying material:

      SKEYID_d = prf(SKEYID, g^xy | CKY-I | CKY-R | 0)
      SKEYID_a = prf(SKEYID, SKEYID_d | g^xy | CKY-I | CKY-R | 1)
      SKEYID_e = prf(SKEYID, SKEYID_a | g^xy | CKY-I | CKY-R | 2)

   and agreed upon policy to protect further communications. The values
   of 0, 1, and 2 above are represented by a single octet. The key used
   for encryption is derived from SKEYID_e in an algorithm-specific
   manner.

   To authenticate either exchange the initiator of the protocol
   generates HASH_I(SIG_I) and the responder generates HASH_R(SIG_R)

where:

    HASH_I = prf(SKEYID, g^xi | g^xr | CKY-I | CKY-R | SAi_b | IDii_b )
    HASH_R = prf(SKEYID, g^xr | g^xi | CKY-R | CKY-I | SAi_b | IDir_b )

SAi_b is the entire body of the SA payload (minus the ISAKMP generic header), all
proposals and all transforms offered by the Initiator.
CKY-I and CKY-R are the Initiator's cookie and the Responder's cookie, respectively,
from the ISAKMP header.
g^xi and g^xr are the Diffie-Hellman public values of the initiator and responder
respectively.
```

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

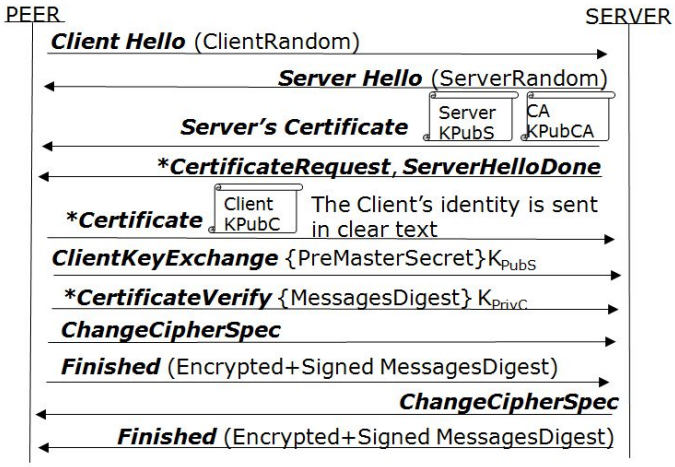


Mode Transport et Tunnel dans le protocole ESP, IPSEC



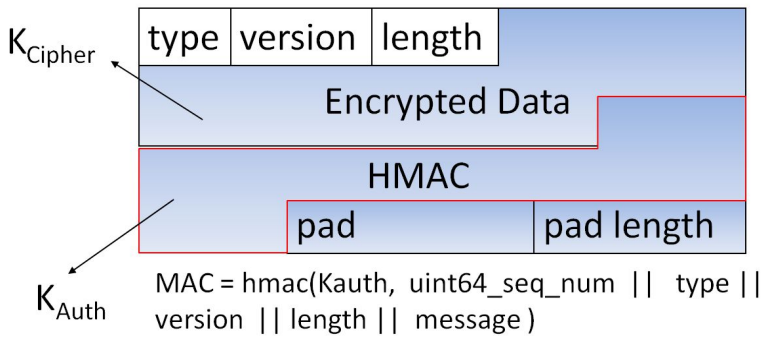
En tête du protocole ESP.

TLS / SSL

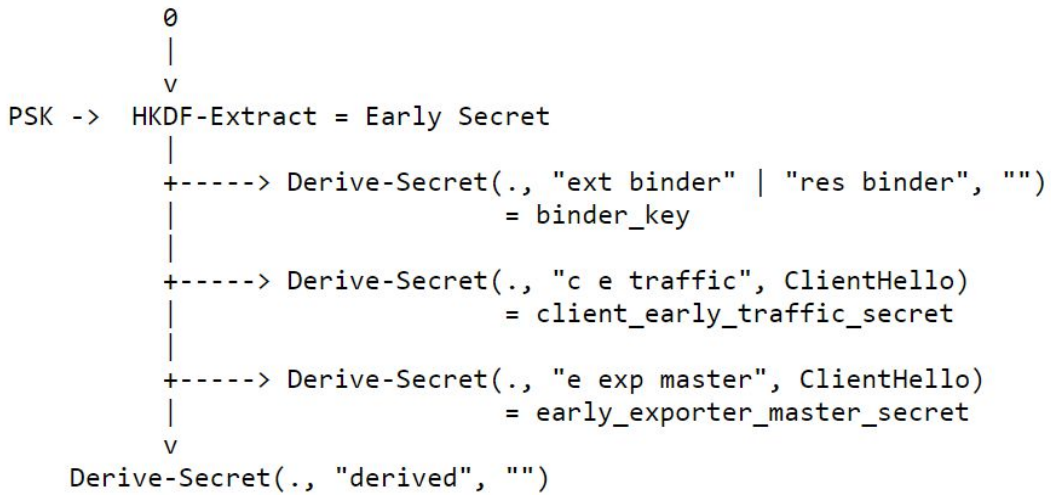


MasterSecret= PRF(ClientRandom, ServerRandom, PreMasterSecret,…)
Keys = PRF(ClientRandom, ServerRandom, MasterSecret,…)

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.



TLS1.3



TLS 1.3 gère trois modes d'échange de clés:

- (EC)DHE: Echange Diffie-Hellman (DH) dans des corps finis ou des courbes elliptiques

- Pre-Shared-Key (PSK) uniquement: clé pré-partagée sans échange DH

- Pre-Shared-Key (PSK) avec (EC)DHE: clé pré-partagée et échange DH

Il existe deux alternatives pour authentifier les serveurs:

- L'usage d'un certificat (message *Certificate*) serveur et d'une clé privée associée, qui calcule une signature (message *CertificateVerify*) permettant d'authentifier l'échange DH.

- La mise œuvre d'un secret pré-partagé (PSK) utilisé dans les procédures de calculs des clés.

Les deux messages en clair sont *ClientHello* et *ServerHello*, chacun comportant une clé publique (aG, bG) permettant de calculer un secret DH abG (en notation additive). Un premier canal sécurisé est établi pour les messages *handshake*. Un deuxième canal est mis en place pour les données applicatives.

TLS1.3 utilise des canaux sécurisés de type *Authenticated Encryption with Associated Data* (AEAD), tels que AES-CCM ou AES-CGM. L'entête du paquet record en clair (soit 5 octets) constitue les données associées authentifiées, la charge est chiffrée et authentifiée.

```
      PSK ->  HKDF-Extract = Early Secret
                |
                +-----> Derive-Secret(., "ext binder" | "res binder", "")
                |                     = binder_key
                |
                +-----> Derive-Secret(., "c e traffic", ClientHello)
                |                     = client_early_traffic_secret
                |
                +-----> Derive-Secret(., "e exp master", ClientHello)
                |                     = early_exporter_master_secret
                v
            Derive-Secret(., "derived", "")
                |
                v
      (EC)DHE -> HKDF-Extract = Handshake Secret
                |
                +-----> Derive-Secret(., "c hs traffic",
                |                     ClientHello...ServerHello)
                |                     = client_handshake_traffic_secret
                |
                +-----> Derive-Secret(., "s hs traffic",
                |                     ClientHello...ServerHello)
                |                     = server_handshake_traffic_secret
                v
            Derive-Secret(., "derived", "")
                |
                v
      0 -> HKDF-Extract = Master Secret
                |
                +-----> Derive-Secret(., "c ap traffic",
                |                     ClientHello...server Finished)
                |                                                     =
client_application_traffic_secret_0
                |
                +-----> Derive-Secret(., "s ap traffic",
                |                     ClientHello...server Finished)
                |                                                     =
server_application_traffic_secret_0
```

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.

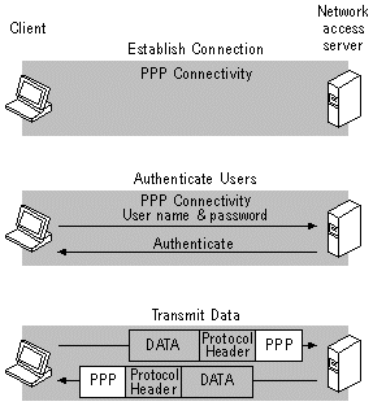
La clé de session (*Session Key*) est dans le cas d'un utilisateur du KDC une clé cryptographique (DES, 3xDES, AES) déduite de son mot de passe; et dans le cas d'un service elle est générée par le KDC.

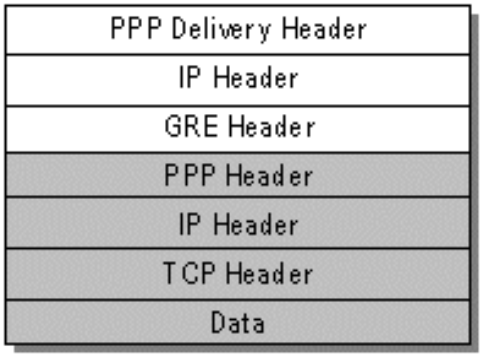
Un *authenticator* est une structure d'authentification comportant l'identité de l'utilisateur et une date chiffrée avec la clé de session service. Un *authenticator* lié à un

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

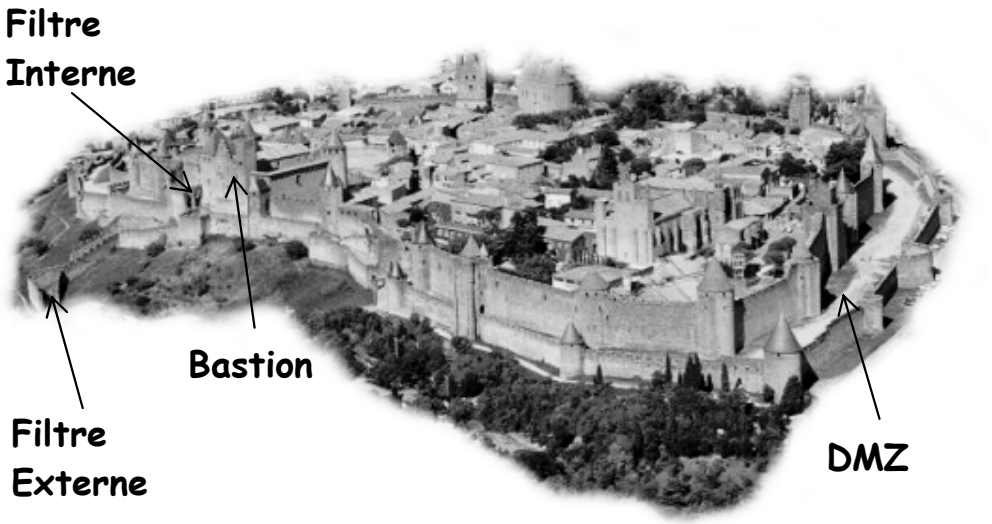
Le document "Protocol Architecture" (RFC 4251) décrit les différents éléments de SSH:

- 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 basé sur un échange Diffie-Hellman, dont les paramètres sont certifiés par une signature DSA généré par le serveur.

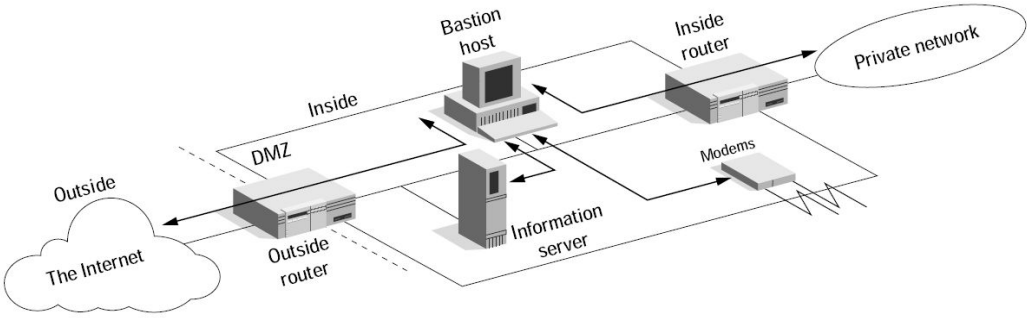
- Le bloc *User Authentication Protocol*, est en charge de l'authentification du client, généralement grâce à un couple login/password, mais il est possible d'utiliser un certificat et la clé privé associée. Les messages sont acheminés via le canal sécurisé mis en place par TLP.

- 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[65] 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 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.

[65] Chuck Semeria, "Internet Firewalls and Security", 1996

Eléments d'attaques

Attaques par "kleptogram" d'échange DH

Capstone était un projet à long terme du gouvernement des États-Unis visant à développer des normes cryptographiques à l'usage du public et du gouvernement. Le projet autorisé par le *Computer Security Act* en 1987, piloté par *le National Institute of Standards and Technology* (NIST) et la *National Security Agency* (NSA), débuta en 1993.

L'initiative impliquait quatre algorithmes standards : l'algorithme de chiffrement Skipjack, la puce cryptographique Clipper implémentant l'algorithme Skipjack, l'algorithme de signature numérique *Digital Signature Algorithm* (DSA), la fonction de hachage SHA-1, et un protocole d'échange de clés. Tous les composants Capstone étaient conçus pour fournir une sécurité de 80 bits.

Le standard NIST SP 800-90A, publié en janvier 1992, recommande l'algorithme "Dual Elliptic Curve Deterministic Random Bit Generation (Dual_EC_DRBG)" , qui contient un backdoor (?) de type *kleptogram*.

Dual EC DBRG

Dual EC DBRG est un générateur de nombre pseudo aléatoire basé sur des courbes elliptiques. L'algorithme est le suivant:

EC une courbe elliptique, par exemple secp256v1 (prime 256), d'ordre n sur Z/pZ

X(x,y)=x, X un point de la courbe EC de coordonnées (x,y)

P_r, Q_r deux points de la courbe

$t(x) = x \bmod p/2^{16}$, x dans Z/pZ

$g_{Pr}(x) = X(x.P_r)$, $g_{Qr}(x) = t(X(x.Q_r))$

$s_1 = g_{Pr}(seed)$, $r_1 = g_{Qr}(s_1)$

$s_k = g_P(s_{k-1})$, $r_k = g_Q(s_k)$, le nombre aléatoire

L'algorithme EC DBRG est un kleptogram.

kleptogram

Le but d'un kleptogram[66][67] est l'attaque d'un algorithme cryptographique tel que DH, par des moyens cryptographiques. L'article définit le kleptogram ci-dessous pour récupérer une clé secrète de DH

1- Définition

Soit un corps Z/pZ, p premier, g un générateur d'ordre q

$Y = g^X \bmod p$, X clé privé de l'attaquant, Y clé publique de l'attaquant

W un entier impair, t un entier (bit) aléatoire de valeur 0 ou 1

a, b entiers constants

H une fonction de hash dans Z/pZ (pseudo aléatoire), H(x) < φ(p) (φ indicatrice d'Euler)

$1< c_1 < p-1$, un nombre aléatoire

Premier message (m_1)

$m_1 = g^{c1} \bmod p$

Deuxième message (m_1)

$z = g^{c1-W.t}\, Y^{-a.c1-b} \bmod p$

c_2= H(z), H(Z): Z/pZ -> Z/pZ

$m_2 = g^{c2} \bmod p$

2- Attaque

En notant $r = m_1{}^a g^b \bmod p$ et $z_1 = m_1 / r^X \bmod p$

Si $m_2 = g^{H(z1)} \bmod p$, alors c_2= $H(z_1)$

En notant $z_2 = z_1 / g^W \bmod p$

Si $m_2 = g^{H(z2)} \bmod p$, alors c_2=$H(z_2)$

En d'autres termes l'observation de deux échanges DH (g^{c1}, g^{c2-}) successifs permet de récupérer la deuxième clé privée (c_2)

3-Démonstration

L'attaque se démontre facilement en développant z sous la forme:

$z = g^{c1-W.t}\, Y^{-a.c1-b} \bmod p = g^{c1-W.t} / g^{X(a.c1+b)} = g^{c1} / g^{W.t} / g^{X(a.c1+b)}$

66 Young, Adam; Yung, Moti (1997-05-11). Kleptography: Using Cryptography Against Cryptography. Advances in Cryptology — EUROCRYPT '97. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. pp. 62–74.

67 Young, Adam; Yung, Moti (1997-08-17). The prevalence of kleptographic attacks on discrete-log based cryptosystems. Advances in Cryptology — CRYPTO '97. Lecture Notes in Computer Science. Springer, Berlin

L'attaquant connait la clé privé X associée à la clé publique Y

Application à la signature DSA

1- Définition de l'algorithme de signature DSA

Z/pZ un corps, p un nombre premier

q un nombre premier qui divise p-1, donc un générateur d'ordre q, $g^q = 1 \bmod p$

Alice: clé privé x < q-1, clé publique $g^x \bmod p$

H une procédure de hash, m un message

k un nombre aléatoire, k < q

$r = (g^k \bmod p) \bmod q$

$s = k^{-1}(H(m) + x^r) \bmod q$

Signature DSA = (r,s)

Bob vérifie $r = \left(g^{s^{-1}H(m)}\, g^{s^{-1}r} \bmod p\right) \bmod q$

2- Utilisation d'un kleptogram pour extraire la clé privée de signature

Un kleptogram permet de connaitre le k de la deuxième signature.

La connaissance de k permet d'extraire la valeur x de la clé privée d'Alice :

$s.k\ /H(m)^{-1} \bmod q = x^r \bmod q$

Soit à résoudre $x^n = a \bmod q$ (q premier)

Si n premier avec q-1, $y = n^{-1} \bmod \varphi(q) = n^{-1} \bmod(q-1)$, $\varphi(q) = q-1$

D'où $x = a^y \bmod q$

Puisque $x^n = a^{yn} \bmod q = a^{1+k(q-1)} \bmod q = a \bmod q$

Il faut donc forger $r = g^{c1}$, r impair, r premier avec q-1

Kleptogram pour courbe elliptique

α, β, ω entiers, ω impair, H pseudo random number generator (notations conformes à l'article[68])

Courbe elliptique d'ordre n, sur le corps Z/pZ, p premier. Q= d.G

1) $c_1 < n$, $M_1 = c_1G$

2) $t \in \{0, 1\}$, $Z = (c_1 - \omega.t)G + (-\alpha.c_1 - \beta)Q$

$c_2 = H(Z)$ (H: EC -> Z/pZ)

$M_2 = c_2G$

[68] Stephan Verbücheln,"How Perfect Offline Wallets Can Still Leak Bitcoin Private Keys", January 2, 2015

Attaque

$R = \alpha M_1$

$Z_1 = M_1 - d.R$

Si $M_2 = H(Z_1).G$ alors $c_2 = H(Z_1)$

$Z_2 = Z_1 - \omega G$

Si $M_2 = H(Z_2).G$ alors $c_2 = H(Z_2)$.

Application à la signature ECDSA

1-Définition

Courbe elliptique (EC) d'ordre n sur le corps Z/pZ, G un générateur (ordre n)

d clé privé, Q = dG clé publique

H une fonction de hash

k un nombre aléatoire $1< k < p$, généralement on choisit $1 < k < n$

$R = (r_1,r_2) = kG$

$r = r_1 \bmod n$

$s = k^{-1} (H(m) + d.r) \bmod n$

la signature est le couple $\sigma = (r,s)$

La clé privée (d) se déduit directement de la valeur de k

$d = (s.k - H(m).r^{-1}) \bmod n$

2- Utilisation d'un kleptogram pour extraire la clé privée de signature

Il est facile de retrouver R= k.G à partir de r. Connaissant r_1 deux points R de la courbe sont possibles (r_1,r_2) et $(r_1,-r_2)$, associés à une clé publique

$Q = r^{-1} (s.R - h(m).G) \bmod n$, permet de vérifier la signature $\sigma = (r,s)$, avec la clé publique R

Un kleptogram permet de connaitre l'entier k de la deuxième signature et donc de calculer la clé publique d'Alice.

Dual EC DBRG est un kleptogram

Si $\beta=0$, $\omega=0$, $Z= c_1.(G -\alpha.Q)= c_1.Pr$, $Q= d.G$, $P_r = G- \alpha.Q$

Pr, Qr deux points de la courbe, le choix de H (EC->Z/pZ) est libre

$s_1 = c_1$

$s_2= g_{Pr}(c_1)= X(s_1.(G-\alpha.Q)) = X(s_1.P_r) = X (c_1.G-\alpha.c_1.d.G)= X(c_1.G-c_1.G.\alpha.d)$

$r_2= c_2 = H(Z) = g_{Qr}(s_2) = t(X(s_2.Q_r))$

Exemple

L'exemple de kleptogram ci-dessous utilise la courbe secp256v1, avec $\beta=0$, $\omega=0$

La fonction t est t(x)= x mod n, et $H(Z) = g_{Qr}(s_2) = t(X(s_2.Q_r))$

PublicKey:
041A8CBAC7870631D60C02EA77A6DF783EF4E715EC592BF358EF93D292519A6C9BB80DCF65F
B3670378852F59F1DD387521B9780B7DE566D45FED61D9EED3D8AE8

PrivateKey:
40FC75E961E681811B22C48755A7D5883511B9D1945736776C4F069666C3F106

Generator:
046B17D1F2E12C4247F8BCE6E563A440F277037D812DEB33A0F4A13945D898C2964FE342E2F
E1A7F9B8EE7EB4A7C0F9E162BCE33576B315ECECBB6406837BF51F5

Order: FFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632551

Prime: FFFFFFFF000000010000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFF

K=c1: 72D9F4541AB515A1802D5158AAD3EED20FBD7DB986F72B5D68D8E75930520A3E

R=k.G: 04
C8EA81D1E5EBDD7840D8046123792ED74EC6A5AC419D9ABDF26A8EF85525D518
9D8A10F64C2850A7DBF6323DC5E525B90797D7AE51436886EFEEA7B31AA1C4FE

Rx: C8EA81D1E5EBDD7840D8046123792ED74EC6A5AC419D9ABDF26A8EF85525D518

r: C8EA81D1E5EBDD7840D8046123792ED74EC6A5AC419D9ABDF26A8EF85525D518

h: 8CD4AE7DA223953050B55B0EE9E4D3830F3B35138A0E1B6CC5F6E6B1E52D1314

s: 051F5890410EF55A37325CD98561330051340AF2929D7F8FDDE787937D00636A

Signature is OK

k.G (from r,s,h):
04C8EA81D1E5EBDD7840D8046123792ED74EC6A5AC419D9ABDF26A8EF85525D5189D8A10F64
C2850A7DBF6323DC5E525B90797D7AE51436886EFEEA7B31AA1C4FE

pubQ: = Q =
0417D3C2C9527C86CE22ABFD853AE1458CBA980795B968143F703ECD02708AD294AFEE98E34
FACF078263B193F2549F3F4044F587EBCEE7E3B2D47C5D07368C5E5

privQ: 01E350D58BF0443235CEA537E7847C8D4AF4B6E793AEC4BCCB22B10427940082

Qr:
04B9A2C4B88BD6C6A011EBF29050E04CC98A049ACA547C21ED494D7712392A4C277224DD60B
514B11ED2FB3DACC465AC187FE16C15684CA5BC4D5CB2C4FA7E4380

alpha: α=
39645BA1F134E2A34FDFE230741AC039ACC42787A0A1DED07F62F37E6DC9C103

Pr: Pr = G-αQ =
046E789967F4B2A39AD748CF165B65213FF7CFA01447EE4071BC1063DCBAA364266BC6DBE61
2FC67C1CBD9531843C98BB1555F5A241F210679723A088649016A0A

Z: Z= c1.(G-αQ) = k.(G-αQ) =
04B34048CF9DBC5BB1AAA81111DC9390BC955AAC36F6F85E19D0BC23CB422A3E5797976FE6F
EC8665623C4A4EF1C03F7FA23A8FE35F8EE17F898F071C10C21CFB1

k1=c2: = H(Z) = (X(Z).Qr) mod n
E1D83B4D0FAB84EEF6232EFE9AEEED16DD709CFBEA86267D12A5BE56684A741B

R=k1.G:
0435D8C116D4310E8634D04884CFF9013F0B7380AE58E6F44EE2C201BC0EF019B9E9E5F0687
AA4E5D064C34181ABCB63616C8E454A43155FB611078D39B06AE2CD

Rx: 35D8C116D4310E8634D04884CFF9013F0B7380AE58E6F44EE2C201BC0EF019B9

r1: 35D8C116D4310E8634D04884CFF9013F0B7380AE58E6F44EE2C201BC0EF019B9

h: E61C21CA716B3B1AEFB7D1198F83679C4CA4D596E5792275DD6203B49216237D

s1: 2477B848294BB9902CAA6518DF4EF524DC758D9CBFA25D2BE16F8F3154492BFA

Signature is OK

k1.G=: (from r1,s1,h):
0435D8C116D4310E8634D04884CFF9013F0B7380AE58E6F44EE2C201BC0EF019B9E9E5F0687
AA4E5D064C34181ABCB63616C8E454A43155FB611078D39B06AE2CD

M1: = k.G =
04C8EA81D1E5EBDD7840D8046123792ED74EC6A5AC419D9ABDF26A8EF85525D5189D8A10F64
C2850A7DBF6323DC5E525B90797D7AE51436886EFEEA7B31AA1C4FE

M2: = k1.G
0435D8C116D4310E8634D04884CFF9013F0B7380AE58E6F44EE2C201BC0EF019B9E9E5F0687
AA4E5D064C34181ABCB63616C8E454A43155FB611078D39B06AE2CD

Z1: = M_1-c_1.α.Q = M_1-c_1.α.d.G = M_1-M_1.α.d
04B34048CF9DBC5BB1AAA81111DC9390BC955AAC36F6F85E19D0BC23CB422A3E5797976FE6F
EC8665623C4A4EF1C03F7FA23A8FE35F8EE17F898F071C10C21CFB1

k1=c2: = H(Z1) =
E1D83B4D0FAB84EEF6232EFE9AEEED16DD709CFBEA86267D12A5BE56684A741B

private key:
40FC75E961E681811B22C48755A7D5883511B9D1945736776C4F069666C3F106

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[69] a proposé dans les années 80 une classification à trois niveaux:

- **La mascarade.** Exploitation illégitime d'un compte utilisateur (cassage de mot de passe …)

- **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.

Les virus sont répartis[70] 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. Ils s'installent dans les secteurs 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.

---

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

70 Vers et Virus, François Paget, DUNOD 2005

Les virus interprétés se propagent dans des macros (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)

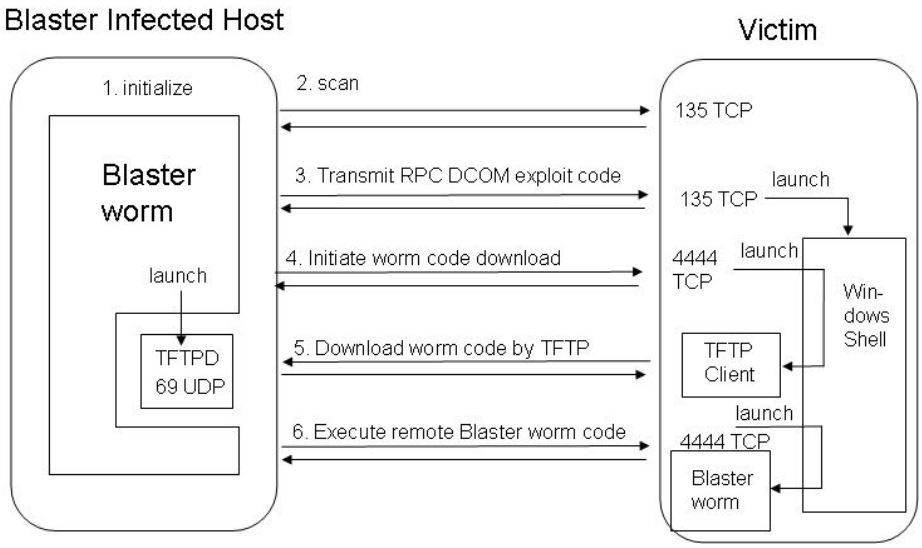
- la recherche générique (recherche de sous ensembles d'instructions)

- la recherche heuristique (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.

Exemple le ver Blaster (2003)



Principe du vers «*Blaster*» (2003)

Au sujet des botnets

Le terme botnet[71] 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.

71 F. Ducrot, M. Danho, X. Marronnier, SÉCURITÉ INFORMATIQUE, numéro 61, CNRS, octobre 2007

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).

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, par exemple pour l'exploitation de "*use-after-free vulnerabilities*".

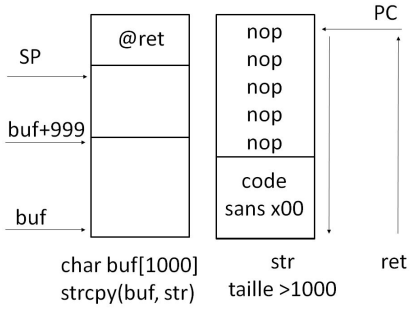
- la *modification de l'adresse de retour du programme* localisée dans la pile, le but étant l'exécution d'un code malveillant contenu dans un paramètre d'appel ou dans une information reçue via le réseau.

- 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.

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[72] "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).

Le *Return-Oriented Programming* (ROP) est une technique d'attaque mettant à profit des fragments de code se terminant par l'instruction return.

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.

[72] "Techniques de Hacking", Jon Erickson, Pearson, 2009-2012

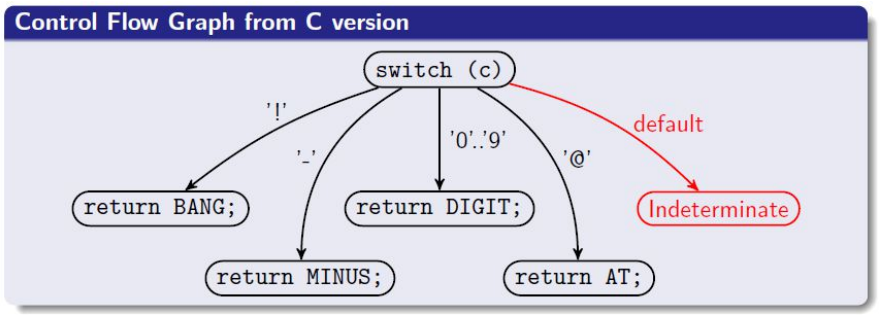
Cross Site Request Forgery CSRF

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.

Obfuscation

L'objectif des techniques d'obfuscation logicielle (soit P un programme, note O(P) sa forme après obfuscation) est de rendre difficile l'analyse d'un programme (selon[73] "Code obfuscation is the practice of making code unintelligible, or at the very least, hard to understand").

Selon le classique paradigme de programmation procédurale, un programme est un ensemble de procédures, d'instructions, de branchements (jump), et de données. Le graphe CFG (*Control Flow Graph*) est une représentation de la structure des appels de procédures et des instructions de saut. Rendre un programme inintelligible consiste à rendre très difficile son analyse (*reverse engineering*) et en particulier l'extraction d'un CFG (voir par exemple la référence[74]). Un mécanisme *deobfuscation* (soit retrouver P à partir de O(P)) peut être un problème NP complet.



Exemple de graphe CFG (*Control Flow Graph*)

Les virus utilisent des techniques polymorphes ou métamorphes.

- Dans le cas du polymorphisme, une clé modifiée à chaque instanciation du programme déchiffre des sous ensembles, qui sont chiffrés ultérieurement avec une nouvelle clé.

-Dans le cas du métamorphisme, le programme polymorphe modifie tout ou partie de son code, qui est recompilé par la suite.

L'obfuscation et la sécurité sont cependant deux notions distinctes. Ainsi après obfuscation, un algorithme cryptographique (DES par exemple) reste exposé aux attaques par injections de faute, SPA (*Single Power Analysis*) ou DPA (*Differential Power Analysis*).

---

[73] A. Balakrishnan and C. Schulze, "Code Obfuscation Literature Survey", http://pages.cs.wisc.edu/~arinib/writeup.pdf, 2005.

[74] Emmanuel Fleury, "Binary Program Analysis: Theory and Practice (what you code is not what you execute)"
http://www-verimag.imag.fr/~async/CCIS/talk_13/Fleury.pdf

Le document[75] "Code Obfuscation Literature Survey" résume les différentes techniques pouvant être mises en œuvre:

- Modification du flux de control du programme (suite de blocs d'instructions en mémoire sans saut, et appels de procédures), répartition aléatoire de blocs d'instructions en mémoire, instructions de saut opacifiées à l'aide de prédicats (par exemple un appel de procédure remplace une instruction jump);

-Modification des abstractions du logiciel, telles que relations d'héritage ou tableaux de données;

- Mise en œuvre d'une machine virtuelle (interpréteur) par le programme, exécutant ses propres instructions (et donc différentes de celles du processeur exécutant le programme);

- Insertion et destruction de code en mémoire au runtime (inline and outline method);

- séparation de variables;

- concaténation de variables;

- conversion de variables en procédures.

L'analyse de programme se décompose en deux classes fonctionnelles:

- L'analyse statique qui examine le code source du programme afin d'en extraire un CFG et certaines informations (des clés cryptographiques par exemple). En code assembleur (binaire) cette opération peut être rendue difficile par l'insertion de zones de données dans le code. En java le code et les données sont séparés dans des zones distinctes du fichier .class. La Dalvik Virtual Machine (DVM) d'Android est similaire à la machine virtuelle Java (JVM), des outils libres publiques ("désassembleur") permettent la transformation de code DVM en JVM, et donc l'analyse statique d'un fichier Android PacKage (APK). Cependant il est possible d'inclure du code C dans une application Android, compilée pour un processeur spécifique (et donc un modèle de mobile donné) grâce aux interfaces JNI (Java Native Interface).

- L'analyse dynamique, réalisée lors de l'exécution du programme (run time).

[75] A. Balakrishnan and C. Schulze, "Code Obfuscation Literature Survey", http://pages.cs.wisc.edu/~arinib/writeup.pdf, 2005.

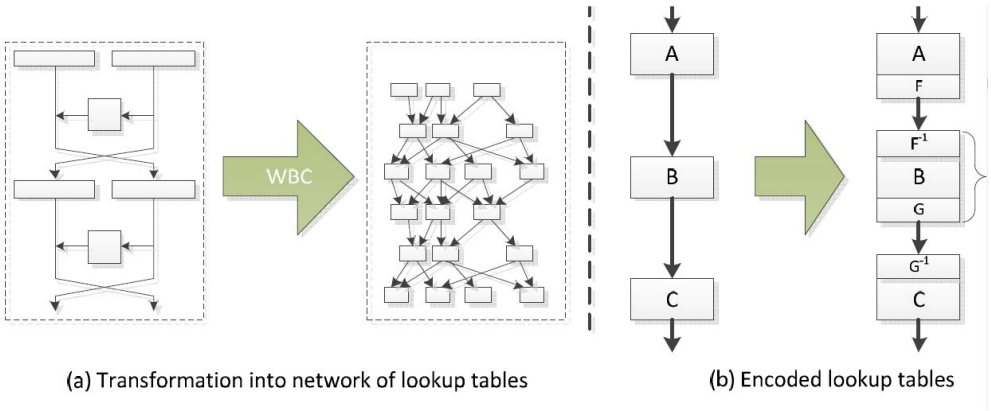
Indistinguishability Obfuscation

L'idée intuitive de l'obfuscation[76] est de construire une boite noire (*Black Box*) qui ne délivre aucune information au cours de son fonctionnement.

En 2001 le célèbre article[77] "*On the (Im)possibility of Obfuscating Programs*" a démontré que le concept de boite noire n'est pas universel, c'est à dire qu'il est toujours possible, quelque soit le procédé d'obfuscation, d'extraire de l'information de certaines fonctions, similaires à des procédures à sens unique (fonction de hash). Cependant le même article introduit la notion d'obfuscation non distinguable (*Indistinguishability Obfuscation*, iO) telle que les obfuscations iO(P1) et iO(P2) de deux variantes fonctionnellement équivalentes d'un programme (par exemple P1= !a + !b, et P2= !a.b) soient indiscernables.

Selon une approche circuit, un algorithme (C1) conçu sous la forme canonique conjonctive (ensemble de AND de termes comprenant des OU de variable logique et de leur négation NOT) serait indiscernable d'une instanciation équivalente C2= iO(C1). Par la suite l'extraction d'information (soit l'obtention de la forme canonique C1 à partir de C2) est impossible en raison du problème bien connu SAT, qui est NP complet. Les algorithmes cryptographiques basés sur le principe iO, sont potentiellement capables de dissimiler certaines informations (clés) mais en sont encore au stade de la recherche.

White-Box Cryptography (WBC)



La technique de White Box a été proposée en 2002[78]. Son objectif[79] 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).

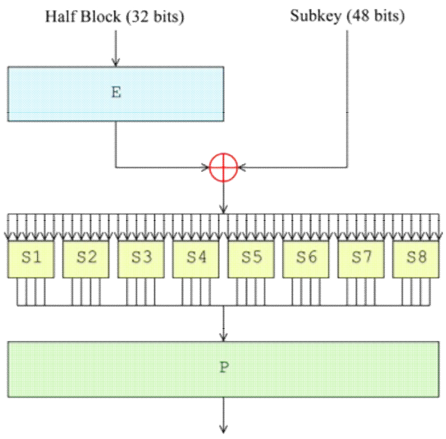
---

[76] Marc Beunardeau ; Aisling Connolly ; Remi Geraud ; David Naccache, "Cdoe Obofsucaitn: Securing Software from Within", IEEE Security & Privacy Issue 3 • May-June 2016

[77] B. Barak et al., "On the (Im)possibility of Obfuscating Programs," Proc. 21st Ann. Int'l Cryptology Conf. (CRYPTO 01), LNCS 2139, Springer, 2001, pp. 1–18

[78] 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.

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



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[80] par des attaques dites DCA (*Differential Computation Analysis*).

Lors de la conférence BlackHat une série d'attaques[81] (injection de fautes et analyse *Differential Fault Analysis* - DFA) a été démontrée sur des implémentations commerciales.

Le WhibOx challenge est une compétition WBC (https://whibox-contest.github.io/), appliquée à l'algorithme AES. Son objectif est la réalisation d'un programme en C, qui calcule un chiffrement AES avec une clé de 128 bits. Le code comporte la fonction :

AES_128_encrypt(ciphertext, plaintext)

Les attaquants disposent du code source et tentent de casser le programme, c'est à dire d'obtenir la clé secrète AES, par tous les moyens disponibles. En 2017 la grande majorité des programmes ont été cassés en moins d'une heure; quelques implémentations ont résistées plus de 10 jours, et le record est d'environ un mois. En 2019 plusieurs implémentations ont résisté pendant la durée du concours

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

[81] Eloi Sanfelix, Cristofaro Mune, Job de Haas, "Unboxing the White Box", BlackHat 2016

Intégrité physique et logicielle

L'intégrité physique et logicielle est un enjeu majeur pour la cybersécurité. Comment avoir une certaine assurance sur l'absence de malware dans un logiciel embarqué, et sur la conformité du hardware (carte électronique et composants) associés ?.

Intégrité physique

Les implants matériels

Un implant est une puce (chip) insérée dans un système électronique, dans le but de réaliser des fonctions malicieuses.

Le 8 octobre 2018 la revue *Bloomberg Business week* a publié un article intitulé "The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies", lequel s'est avéré par la suite totalement infondé. L'article décrit une puce électronique (un implant) de la taille d'une pointe de crayon, soudée sur une carte mère destinée à des environnements cloud, et embarquant des ressources réseau

Cependant en 2020 lors de la conférence CS3sthlm[82] le chercheur Monta Elkins a

présenté dans sa communication "*Nation-State Supply Chain Attacks for Dummies and You Too*" un implant basé sur le processeur ATTINY85 (8Ko FLASH, 512 octets EEPROM, 512-octets SRAM,). Ce boitier de petite taille (4x4x0.8mm) monté sur un module (DigiSpark) coute 1$ il est programmable avec l'IDE Arduino. L'implant est soudé sur la carte mère du routeur CISCO ASA5505. Le principe de l'attaque consiste à injecter lors du boot, des commandes sur l'interface série disponible pour l'administration du dispositif. Le micro logiciel crée un compte SSH permettant l'accès au routeur à distance et signale cette opportunité par une série de ping sur un serveur de l'attaquant.

D'autres idées d'implant visent les claviers, par exemple le projet Haunted USB Cable[83]) (2010) utilise une puce ATTINY45 insérée dans une prise USB, qui réalise l'émulation d'un clavier USB HID (*Human Interface Device*). Le micro logiciel modifie de manière aléatoire les modes CapsLock, Insert, et NumLock, et injecte des caractères espace, backspace, tab, or delete.

En 2016 le projet[84] a adapté le micro logiciel *Haunted USB* pour un processeur ATTINY85. La même année un autre projet[85] a développé une variante logicielle qui injecte une série de caractères après trois pressions de la touche CapsLocks. Les claviers sont équipés de trois voyants indiquant l'activation des modes scroll lock, caps lock et insert. Le système d'exploitation envoie des messages USB HID pour

82 https://www.youtube.com/watch?v=WTncGSYSbNY
83 http://imakeprojects.com/Projects/haunted-usb-cable
84 https://github.com/whiteneon/haunted-usb-1.0
85 https://github.com/Uxio0/usbAspKeyboard

synchroniser les affichages de plusieurs claviers. La séquence d'échange est la suivante:

1) la touche caps lock est pressée; 2) le micro logiciel du clavier notifie cet évènement dans un "input report"; 3) le PC diffuse un message "output report" pour l'activation de la LED correspondante; 4) le micrologiciel active la LED.

Sous Windows des API dédiées permettent la gestion des différents modes d'un clavier. Il est donc possible d'établir un canal caché avec un malware HID USB.

Certains modules Arduino, par exemple basés sur le processeur ATMEGA32u4 supportent des librairies HID réalisant des interfaces clavier et souris. L'ajout d'un SoC de communication, tel que Bluetooth ou Wi-Fi, permet d'injecter des messages indiquant le déplacement du curseur ou la frappe de touche. Typiquement l'attaque consiste à démarrer un *shell* pour exécuter des commandes.

Les relais

Les attaques par relais (*Relay Attack*) s'appuient sur une technique de type MIM (*Man In the Middle*) dont le but est d'étendre la distance de fonctionnement, et/ou de transmettre partiellement les informations reçues.

Par exemple on peut relayer des signaux radio; des attaques SARA (*Signal Amplification Relay Attacks*) sont été constatées pour le vol de voiture équipées de systèmes RKS (*Remote Keyless System*). Une clé électronique détecte en permanence les signaux émis à courte portée par les véhicules, qui sont déverrouillés au terme d'un échange d'authentification.

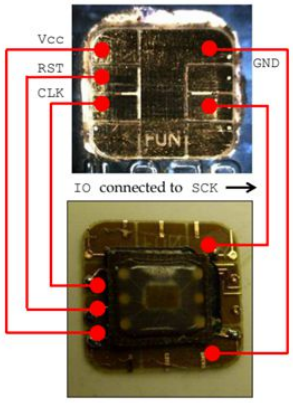
En 2005 un article de Gerhard Hancke[86] démontrait l'usage à distance de carte MIFARE largement déployé pour le contrôle d'accès. Le relais transporte le contenu de trames MAC à travers internet. Les temps de réponse sont plus importants, ce qui est détectable par des contre mesures. Selon le même principe, des attaques sont possibles entre un terminal de paiement et une carte bancaire sans contact, au moyen d'un relais des commandes ISO7816, dites APDUs.

En 2010 une équipe de chercheurs[87] de l'Université College London et de l'université de Cambridge ont démontré une faille de sécurité dans le protocole EMV. L'évènement "*Code PIN non renseigné*" n'est pas mémorisé dans le cryptogramme de paiement, ce qui rend possible une attaque par relais. Ils ont développé un système électronique qui transmet toutes les commandes ISO7816 à la puce EMV authentique, sauf la commande VerifyPIN, pour laquelle le relais délivre la réponse ISO7816 0x9000 (soit PIN OK). Cette faille rend possible l'usage de cartes volées.

[86] Gerhard Hancke "A Practical Relay Attack on ISO 14443 Proximity Cards", 2005

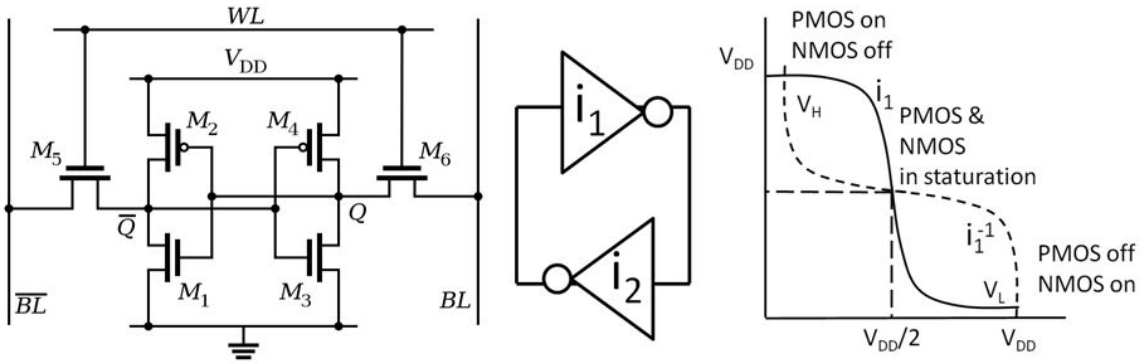
[87] Steven J. Murdoch, Saar Drimer, Ross Anderson, Mike Bond, "Chip and PIN is Broken", 2010 IEEE Symposium on Security and Privacy.

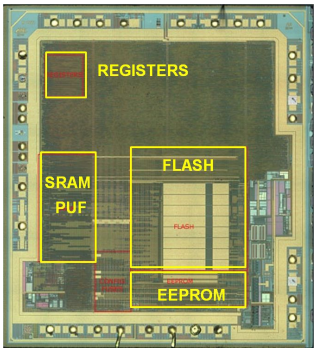
En mai 2011, le GIE Cartes Bancaires a constaté qu'une dizaine de cartes EMV, volées en France quelques mois auparavant, étaient utilisées en Belgique. Une enquête policière a été ouverte, des cartes frauduleuses ont été saisies et expertisées. L'article[88] présente les résultats de cette expertise mettant en évidence l'usage d'un système relais nommée FUN, comportant une mémoire EEPROM 8Ko série (AT24C64), et le processeur AT90S8515A (512o EEPROM, 512o RAM). FUN est une version très intégrée du prototype de l'équipe anglaise; c'est un en fait un implant collé sur le module EMV authentique.



Anti clonage de processeur : techniques SRAM PUF

Une puce électronique peut être remplacée par un clone aux fonctionnalités identiques, ou par un système réalisant son émulation. Les techniques PUF[89] (*physical unclonable function*) permettent d'identifier une mémoire SRAM de manière pseudo unique.





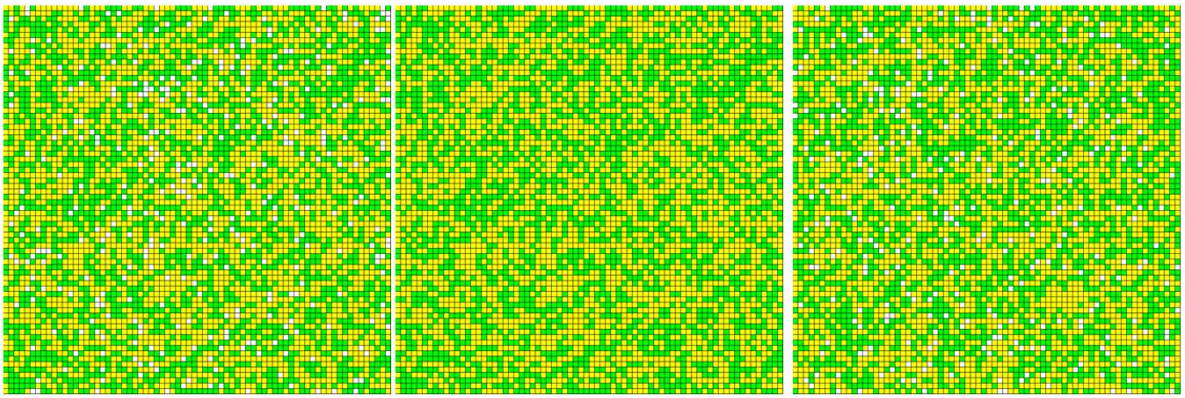
Un microcontrôleur (MCU) comporte une unité arithmétique et logique (ALU) des mémoires non volatiles (EEPROM, FLASH) et éphémères (SRAM). Le rapport de surface entre mémoire FLASH et SRAM est d'environ 4 (?).

Une cellule SRAM classique est réalisée par 6 transistors CMOS, réalisant deux inverseurs connectés en série afin de provoquer un état bistable. Idéalement pour des faibles valeurs de tension d'alimentation VDD les sorties des inverseurs restent proches de VDD/2, avec un gain croissant. Si les inverseurs sont parfaitement symétriques, la tension de sortie suit VDD/2, et bascule de manière aléatoire (avec une probabilité de 50%) vers 0 ou

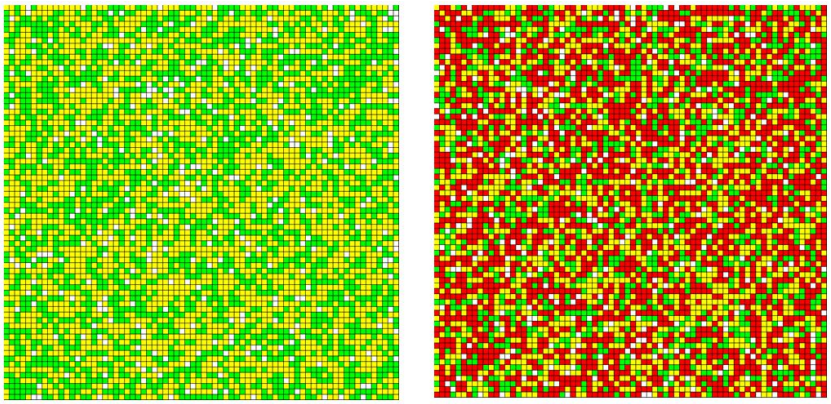
[88] Houda Ferradi, Rémi Géraud, David Naccache, and Assia Tria, "When Organized Crime Applies Academic Results A Forensic Analysis of an In-Card Listening Device", October 2015, Journal of Cryptographic Engineering.

[89] Charles Herder, Meng-Day Yu, Arinaz Koushanfar, Rinivas Devadas, "Physical Unclonable Functions and Applications: A Tutorial", in IEEE Volume 102, Issue: 8, Aug. 2014.

VDD. Cependant en fonction de dissymétries géométriques ou électriques des transistors CMOS, une cellule mémoire peut prendre une valeur fixe (0 ou 1) de manière reproductible (avec une probabilité très proche de 1). Cet effet peut être mis à profit pour identifier une mémoire SRAM intégrée dans un MCU.



La figure ci-dessus monte l'état d'une mémoire SRAM pour deux MCUs ATMEGA8 (à gauche et à droite) après 250 mises en tension. Les carrés blancs indiquent les cellules mémoires bruitées (c'est-à-dire observées avec des valeurs de sorties différentes). Au centre elle montre une image mémoire pour une mise en tension unique.



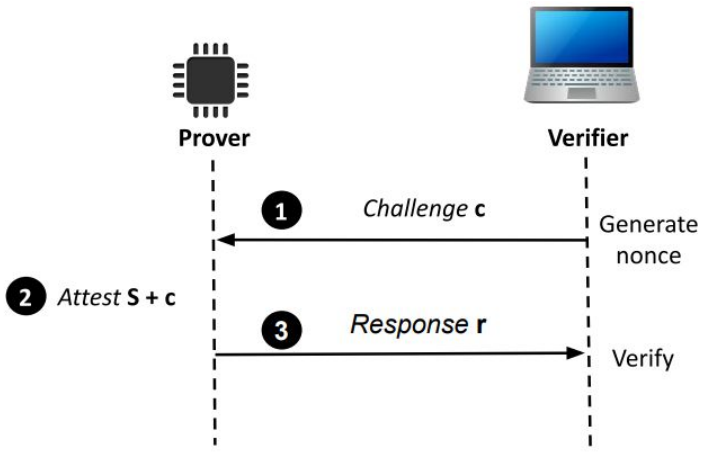
La figure ci-dessus montre les contenus de cellules mémoires différents (en rouge) pour deux processeurs (les points bruités en blanc sont ignorés); elle démontre l'authentification (statique) de SRAM par un procédé PUF. On peut utiliser des cellules SRAM avec des contenus invariants, et protégées d'éventuelles erreurs par des codes auto correcteurs, comme des clés privées, par exemple de taille 256 bits. Cependant les calculs cryptographiques associés permettent de récupérer ces clés, si des techniques de canaux cachés sont pertinentes.

Attestation distante (remote attestation)

L'attestation[90] est une procédure consistant à démontérer (à un évaluateur) une propriété (claim) d'une *cible*, à l'aide de preuves démontrant cette propriété.

[90] George Coker, Joshua Guttman, Peter Loscocco1, Amy Herzog,Jonathan Millen, Brian O'Hanlon, John Ramsdell,Ariel Segall, Justin Sheehy, and Brian Sniffen, "Principles of Remote Attestation"

L'attestation à distance[91] est un mécanisme qui permet à un évaluateur (*Verifier*) de vérifier l'intégrité d'un système hôte (distant) qui est dans un état connu, selon un mécanisme de défi-réponse. Les principales entités d'un tel protocole sont le *Prover* (P) et le *Verifier* (V). Typiquement le *Verifier* produit un nombre aléatoire (nv), le *Prover* génère des données d'attestation (att_data(nv)) et retourne cette information optionnellement munie d'une signature cryptographique.



On distingue trois classes d'attestation:

- Attestation logicielle, le mécanisme d'attestation repose sur des mesures de temps strictes.
- Attestation matérielle, la procédure d'attestation utilise un composant matériel durci ("*tamper-resistant*", tel que TPM par exemple).
- Attestation hybride, la procédure d'attestation utilise des ressources logicielles et matérielles. Par exemple un code stocké dans une mémoire ROM réalise le Prover.

Les attaques[92] logicielles possibles sont par exemples

1. Le pré-calcul de tous les résultats d'attestation, indépendamment du défi.

2. Le rejeu d'un résultat, ou la réponse d'un autre dispositif.

3. La copie mémoire totale (*Memory Copy*).

4. La substitution de données (copie partielle).

5. La Compression.

6. L'usurpation d'identité, envoi d'une réponse invalide ou l'usage de dispositifs valides.

7. La technique ROP (*Return Oriented Programming*). Exécution de fragment de code.

---

91 IOANNIS SFYRAKIS and THOMAS GROSS,"A Survey on Hardware Approaches for Remote Attestation in Network Infrastructures"

92 Sigurd Frej Joel Jorgensen Ankergard, Edlira Dushku and Nicola Dragoni, State-of-the-Art Software-Based Remote Attestation:Opportunities and Open Issues for Internet of Things

Des attaques matérielles peuvent effacer un dispositif authentique, utiliser un dispositif authentique localement (implant) ou à distance (relais).

Intégrité logicielle pour un système embarqué

Un logiciel peut il prouver son intégrité dans un contexte de système embarqué ?

Considérons un microcontrôleur équipé de mémoires non volatile et éphémères. On peut remplir la mémoire d'instructions non utilisée par un content pseudo aléatoire pour éviter les attaques de type "memory copy", et de même pour la mémoire SRAM non utilisée pour les données. L'utilisation de procédure de compression/décompression augmente le temps de calcul. L'idéal serait de concevoir un algorithme optimal en termes de taille mémoire et de temps d'exécution[93]. Les invariants mis en œuvre sont la taille finie des mémoires et le temps. Voici deux exemples illustrant la preuve d'intégrité de logiciel.

Remote Attestation

```
//Input: y number of iterations of the verification procedure
//Output: Checksum C
//Variables:
//       [code_start, code_end]: verified memory area
//       daddr: address of current memory access
//       b: content at daddr
//       x: value of T function
//       l: loop counter
//       SR: status (flags) register
for l = y to 0 do
   //T function updates x where 0 < x < 2^16
   x ← x + (x^2 ∨ 5)  mod 2^16
   //Compute random memory address using x
   daddr = ((daddr ⊕ x) ∧ MASK) + code_start
   //Calculate checksum. j: current index into checksum vector.
   C_j ← C_j + PC ⊕ mem[daddr] + l ⊕ C_{j-1} + x ⊕ daddr +
         C_{j-2} ⊕ SR
   C_j ← rotate left(C_j)
   //update checksum index
   j ← (j + 1)  mod 10
end for
```

L'attestation à distance ("*remote attestation*") est un processus dans lequel une entité de confiance ("*verifier*") mesure à distance l'état interne d'un système potentiellement compromis, à l'aide d'un programme dit "*prover*". L'idée est de réaliser un checksum ou un hash sur les mémoires du système, dans un ordre pseudo aléatoire, fixé par une permutation.

Par exemple l'article[94] utilise une permutation[95] P dans $Z/2^n$, c'est-à-dire pour un espace d'adresses de 2^n cellules.

$$P(x) = x + x^2 \vee C \mod 2^n$$

Les bits 0 et 3 de C doivent être à 1, soit C = 5 v X mod 2^n L'opération carré est réalisé par un multiplicateur câblé, ce qui garanti un temps d'exécution minimal. L'algorithme *Prover* calcule un checksum des mémoires du système, le nombre d'itération étant aléatoirement fixé par le vérificateur, à l'aide d'une implémentation supposée optimale en temps d'exécution.

---

93 Asokan, N. et al. "ASSURED: Architecture for Secure Software Update of Realistic Embedded Devices.". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 37.11 (2018): 2290-2300.

94 Seshadri, A. et al. "SCUBA: Secure Code Update By Attestation in sensor networks.", in Radha Poovendran & Ari Juels, ed., "'Workshop on Wireless Security" , ACM, , pp. 85-94 (2006).

95 A. Klimov, and A. Shamir. "A New Class of Invertible Mappings.", . CHES, volume 2523 of Lecture Notes in Computer Science, page 470-483. Springer, (2002)

Bijective MAC, BMAC

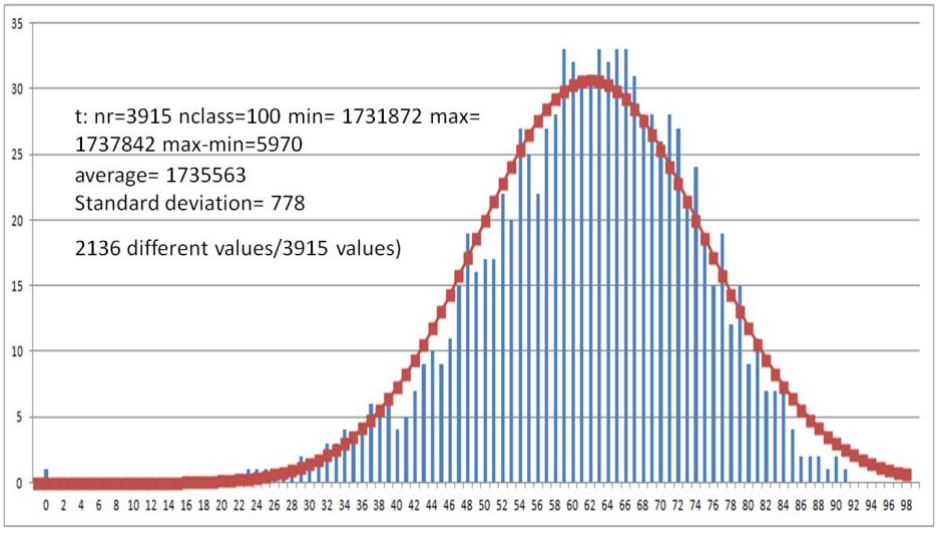
L'algorithme BMAC[96] réalise une empreinte avec une fonction à sens unique (SHA2, SHA3…) sur un espace mémoire de taille m, et en utilisant une permutation P.

$$bMAC(P) = h(\, A(P(0) \,||\, A(P(1)|| \ldots || \, A\big(P(i)\big) \ldots || \, A(P(m-1)\,)$$

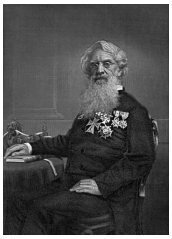
Soit un nombre premier p >m, la permutation P(y) = F(1+y) -1 est basée sur des générateurs deux générateurs (g1, g2) et un éléments (s1) dans Z/pZ^*.

$$F(x) = \; {g_2}^{s_1 {g_1}^x \, mod \, p} mod \; p, \;\; x, s_1 \in [1, p-1]$$

Lorsque l'implémentation de F(x) utilise l'algorithme *square & multiply*, on observe une distribution normale pour les temps de calcul.



Les Canaux Cachés.



Bien que les algorithmes cryptographiques tels que RSA ou DES soient considérés comme sûrs 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 - *Simple Power Analysis* & DPA – *Differential Power Analysis*, Paul Kocher).

[96] P. Urien, "Time Stamped Bijective MAC and Dynamic PUF Authentication New Directions For IoT Security : Invited Paper," 2020 Sixth International Conference on Mobile And Secure Services (MobiSecServ), Miami Beach, FL, USA, 2020

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(forme chiffrée) = M^e modulo m = M x M x ….x M (e fois)

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_i modulo m, si ei=1; M_0 = M, $M_i = M_{i-1} \times M_{i-1} = M^{2\wedge i}$

En constate que, dans cette implémentation de l'algorithme RSA (dite *square and multiply*), 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 des 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):

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

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, alors la moyenne de leur produit est égal au produit des moyennes, et donc est proche de 0.

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

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.

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

Quelques Attaques

Faites moi confiance ? 20 ans de bugs et heuristiques

Voici une liste non exhaustive de failles de sécurité.

- 1995, Peter Shor[97] 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 une 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[98] 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) d'essais par l'université de Berkeley[99].

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

- 2002, trois scientifiques indiens[101] Manindra Agrawal, Neeraj Kayal et Nitin Saxena publient le test de primalité AKS, qui a reçu le prix Gödel en 2006. La

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

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

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

[100] 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.

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

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*.

- 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[102], mettent en évidence l'absence de vérification du code PIN dans certaines cartes bancaires EMV, lors de transactions de paiement.

- En 2010, une présentation au *27ième Chaos Communication Congress* à Berlin ("Console Hacking 2010, PS3 Epic Fail") décrit une attaque permettant d'obtenir la clé de signature ECDSA utilisée par les consoles SONY PS3. Cette dernière est basée sur le fait que le logiciel utilise un nombre aléatoire fixe !.... Quelques rappels sur la signature la signature ECDSA:

n est l'ordre (premier) d'un groupe définit sur une courbe elliptique

Soit $x \in [1,n-1]$ une clé privé, $P= xG$ (G un générateur) est la clé publique

k un nombre aléatoire $k \in [1, n-1]$

$kG= (x_R,y_R)$, et $r = x_R \bmod n$

size = nombre d'octets de n, size=32 pour une courbe elliptique de 256 bits.

H une fonction de hash, e=H(M), M un message. Si H(M) produit plus de size octets, on choisit les size octets les plus à gauche (octets de poids fort).

Le couple (r,s), avec $s = k^{-1} (e + x r) \bmod n$ est la signature du message M.

Soit deux signatures (r,s1) et (r,s2), e1=h(M1), e2=h(M2) avec le même r.

La clé privée s'écrit $x = (e1\ s2 - e2\ s1)\ r^{-1}\ (s1-s2)^{-1} \bmod n$

- 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[103], 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

[102] 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

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

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*".

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

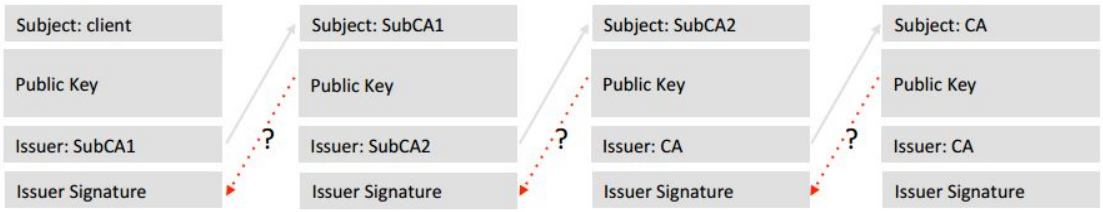
- En 2013, un document publié par Edward Snowden révèle que les échanges de données entre les Data Center de la société Google ne sont pas chiffrés.

- En 2014 un groupe[104] 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 $\mathbb{F}_Q = \mathbb{F}_{q^{2k}}$ avec k#q

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

- En 2014 Jeff Forristal directeur technique de la société Bluebox a publié lors de la conférence[105] 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.



- 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.

---

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

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

```
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[106] 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[107] 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 résumée par la relation ci dessous avec $N=2^p$ et k compris entre 1 et 2.

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

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

- En 2016 le FBI demanda à APPLE, qui déclina la requête, de rendre possible l'accès à des données chiffrées dans une iphone5 impliqué dans l'affaire dite de "*l'attaque de San Bernardino*". Le système d'exploitation iOS calcule les clés de chiffrement des fichiers à partir du passcode de l'utilisateur, et de divers éléments stockés dans le smartphone. Le nombre d'essais quant au renseignement du

106 "BadUSB On accessories that turn evil"; KarstenNohl; Sascha Krißler; Jakob Lell.
107 "Imperfect Forward Secrecy:How Diffie-Hellman Fails in Practice" David Adrian et Al.

passcode est limité, et en cas d'échec les données sont effacées. L'article[108] de Sergei Skorobogatov décrit le hack qui permet de contourner ces défenses, selon une technique dite *copie d'image mémoire* ("NAND mirroring"). La mémoire NAND du mobile est extraite puis dupliquée. Cette copie comporte également la lecture de blocs cachés. L'iphone5 est modifié pour être équipé d'une mémoire NAND externe. L'attaque consiste à essayer tous les passscode possibles, en restituant la mémoire NAND initiale tous les 5 essais.

- 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[109], chercheur à l'université californienne de Berkeley, met en évidence un trou de sécurité de l'application WhatsApp[110], basé sur une technique MIM (Man In the Middle) possible dans des réseaux GSM.

- En 2017 un groupe de chercheurs[111] casse la sécurité des ampoules connectées HUE. Ils démontrent qu'il est possible de les reprogrammer à partir d'une voiture ou d'un drone, et donc de propager un malware informatique à l'échelle d'une ville. L'attaque se déroule en deux temps. Une commande unicast de réinitialisation usine, force l'ampoule à rejoindre le réseau ZigBee de l'attaquant. Chaque mise à jour du micrologiciel est chiffrée et authentifiée par l'algorithme AES-CCM. Cependant, toutes les ampoules utilisent la même clé, qui a été récupérée par une attaque CPA (*Correlation Power Analysis*).

- En mai 2017 le vers WannaCry (encore dénommé WannaCrypt), a infecté plus de 300000 ordinateurs, répartis dans plus de 150 pays. Il est considéré comme le plus grand piratage à rançon (*ransonware*) de l'histoire d'Internet.

Le malware est basé sur la faille Microsoft "*Windows SMB Server CVE-2017-0145 Remote Code Execution Vulnerability*", découverte le 14 mars 2017. Cette faille est associée au message SMB TREE_CONNECT Response (un buffer overflow ?)

Un exploit logiciel installe le vers *EternalBlue*, qui télécharge par la suite un ensemble de composants logiciels. Le mécanisme de propagation réalise un scan de port TCP 445 (Server Message Block, SMB), puis la détection du bug MS17_010, et enfin l'injection de l'exploit *Eternal Blue - Double Pulsar*. Une URL utilisant un nom de

---

108 Sergei Skorobogatov, "The bumpy road towards iPhone 5c NAND mirroring", 2016

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

110 WhatsApp Encryption Overview, Technical white paper November 17 2016

111 Eyal Ronen, Colin O'Flynny, Adi Shamir and Achi-Or Weingarten, "IoT Goes Nuclear: Creating a ZigBee Chain Reaction", (2017)

domaine à priori non alloué, réalise une sonde (*Kill Switch*) de détection de machine virtuelle.

Le vers génère une paire de clés RSA (en s'appuyant sur la bibliothèque *Windows Crypt*); des fichiers sont ensuite chiffrés à partir de la clé publique. Le ransomware utilise trois adresses Bitcoin pour collecter les paiements:

- 115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn
- 12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
- 13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94

- En 2019 un groupe de chercheurs[112] a démontré une attaque qui permet au moyen d'un laser de produire un son sur le microphone MEMS des assistants personnels. Il est possible d'utiliser des commandes vocales pour des distances relativement importantes (100m).

- SIM SWAPPING 2020. Le SIM swapping consiste à usurper un numéro de mobile. Le pirate demande à activer un numéro sur une carte SIM en sa possession. Pour convaincre le service client de l'opérateur, il utilise des informations personnelles (date de naissance, adresse, numéro de client, etc.), qu'il a trouvé par ailleurs. Par la suite il peut émettre des appels téléphoniques ou des SMS.

- Ripple20[113]. L'entreprise Treck a développé à partir des années 1997 une librairie TCP/IP; par la suite elle a collaboré avec la société Elmic Systems pour de nouvelles versions de cette pile de communication. Cette dernière est intégrée dans de nombreux (de l'ordre du milliard) dispositifs électroniques par exemple dans le domaine de la santé. En 2020 le laboratoire *JSOF research lab* a identifié 19 vulnérabilités[114] Zero-Day permettant par exemple l'exécution de shell code.

- SolarWinds 2020. SolarWinds est un éditeur de logiciels (basé au Texas), dédiés à la surveillance (*monitoring*) de réseaux (la plateforme Orion). Ces produits sont utilisés par des grands groupes tels que: Microsoft ou Amazon mais également le *Department of Homeland Security* (DHS). Un cheval de Troie[115] a été injecté dans une

---

112 Takeshi Sugawara, Benjamin Cyr, USara Rampazzi, Daniel Genkin, Kevin Fu "Light Commands: Laser-Based Audio Injection Attacks on Voice-Controllable Systems", 2019

113 https://www.jsof-tech.com/wp-content/uploads/2020/06/JSOF_Ripple20_Technical_Whitepaper_June20.pdf

114 https://i.blackhat.com/USA-20/Wednesday/us-20-Oberman-Hacking-The-Supply-Chain-The-Ripple20-Vulnerabilities-Haunt-Tens-Of-Millions-Of-Critical-Devices.pdf

115https://www.microsoft.com/security/blog/2020/12/18/analyzing-solorigate-the-compromised-dll-file-that-started-a-sophisticated-cyberattack-and-how-microsoft-defender-helps-protect/

mise à jour, et détecté en décembre 2020. C'est une attaque qualifiée de *Supply Chain Attack*.

- L'article[116] présenté lors de la conférence CHESS 2019, démontre le clonage de clés électroniques utilisées par un véhicule TESLA. L'attaque met à profit les failles de sécurité de l'algorithme de signature cryptographique DST40. La clé secrète (40 bits) est extraite grâce à l'enregistrement de deux séquences défi (40bits)/signature (24 bits).

- L'article[117] démontre la récupération d'une clé privé utilisé par les jetons FIDO U2F, Google Titan Security Key. La procédure d'authentification est une signature ECDSA sur la courbe P256. Une attaque par canaux caché (DPA) est réalisé avec succès sur l'algorithme "*comb*" qui réalise une multiplication scalaire pour un nombre de 256 bits (clé privée) en 128 opérations.

- TLS Raccoon Attack. L'article[118] démontre une attaque par canaux caché, de type *timing attack*, qui permet de récupérer la clé privée utilisée pour une échange de Diffie-Hellman. Elle suppose une clé publique (g^b) fixe côté serveur avec une taille de l'ordre de 1024 bits. L'attaquant (le client) génère des clés $g^a x g^{ri}$, impliquant la construction de secret partagé g^{ab} x $g^{ri\ b}$. Les versions de TLS 1.2 et inférieures calculent le pre-master-secret en réalisant une procédure de hash sur le secret DH. Cependant les octets nuls de poids fort sont ignorés, ce qui conduit un calcul sur un nombre binaire de longueur variable. Certaines valeurs produisent donc des temps calculs plus petits. L'article estime à 20 millions le nombre d'essais nécessaires à l'extraction d'une clé privée de 1024 bits.

- TLS Selfie Attack. Le protocole TLS1.3 peut utiliser un secret pré-partagé (pre-share-key, PSK) pour authentifier les échanges de Diffie-Hellman entre client et serveur. L'article[119] démontre un nœud internet qui réalise la réflexion des paquets TLS vers l'adresse source (celle du client). Si le nœud d'origine exécute également un serveur TLS1.3 avec le même PSK, celui-ci reçoit des messages d'un client TLS1.3 PSK supposé distant (l'en tête TCP/IP n'est pas liée de manière cryptographique à la charge TLS).

---

116 Lennert Wouters, Eduard Marin, Tomer Ashur, Benedikt Gierlichs, Bart Preneel, Fast, Furious and Insecure: Passive Keyless Entry and Start Systems in Modern Supercars, CHESS Conference 2019

117 Victor Lomne and Thomas Roche, A Side Journey to Titan Side-Channel Attack on the Google Titan Security Key (Revealing and Breaking NXP's P5x ECDSA Implementation on the Way), Janvier 2021

118 Robert Merget, Marcus Brinkmann, Nimrod Aviram, Juraj Somorovsky, Johannes Mittmann, and Jörg Schwenk, Raccoon Attack: Finding and Exploiting Most-Significant-Bit-Oracles in TLS-DH(E), Février 2021

119 Nir Drucker and Shay Gueron, Selfie: reflections on TLS 1.3 with PSK, 2019

- Attaque EMV 2020. Les références[120] [121] [122] démontrent la réalisation de transactions EMV sans la présentation de code PIN. L'attaque repose sur deux faiblesses cryptographiques du protocol EMV. Le kernel 3 utilisé par le standard *Visa qVSDC* (aussi dénommé *Visa payWave*), permet la négociation en fonction du montant de la transaction et du type d'application EMV, selon deux paramètres TTQ (*Terminal Transaction Qualifiers*, envoyé à la carte) et CTQ (*Card Transaction Qualifiers*, généré par la carte). Certaines applications EMV hébergées par exemple sur des smartphones Android, gèrent la présentation du PIN sur le mobile. L'échange des attributs TTQ/CTQ n'est pas protégé par des moyens cryptographiques. Une attaque de type relais (*Man In the Middle*) permet d'indiquer au terminal la (fausse) vérification du PIN par le mobile. La deuxième partie de l'attaque consiste à maquiller (à l'aide d'un relais) une application EMV (par exemple Mastercard) pour la présenter selon le profil *Visa qVSDC*. Cette opération est rendu possible par le fait que la lecture de la liste des applications hébergées dans la carte de paiement (stockée dans l'application PSE, *Payment System Environment*.) n'est pas protégée

- Phantom of the ADAS, 2020. L'article[123] décrit une attaque sur le système de conduite autonome des TESLAs (Autopilot – Full Self Driving FSD). Des images sont projetées durant une courte période (50 ms), et donc sont imperceptibles par un humain. La détection par les caméras embarquées de panneaux fantômes, tels que stop, limitation de vitesse, ou d'obstacles entrainent des arrêts rapides ou des comportements dangereux. Une contremesure est proposée basée sur un réseau à convolution.

- Juillet 2022, l'algorithme post quantique SIKE, candidat à la standardisation par le NIST, est cassé[124] par une équipe de recherche de l'université de Louvain, en 62 minutes sur un processeur Intel Xeon E5-2630v2 à 2,60 GHz.

---

120 David Basin, Ralf Sasse, and Jorge Toro-Pozo, The EMV Standard: Break, Fix, Verify, 42nd IEEE Symposium on Security and Privacy (S&P), 2021

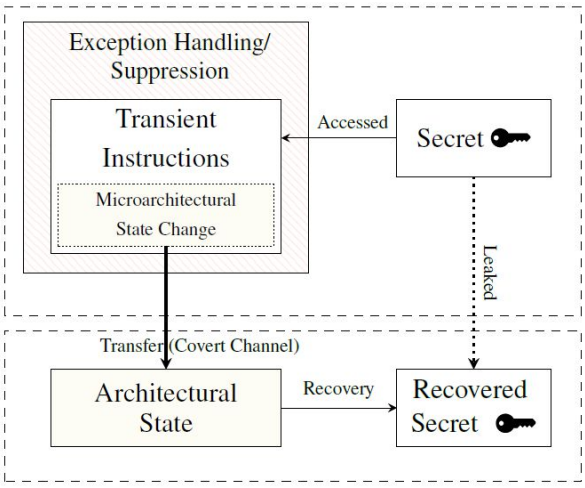
121 David Basin, Ralf Sasse, and Jorge Toro-Pozo, Card Brand Mixup Attack:Bypassing the PIN in non-Visa cards by Using Them for Visa Transactions,
30th USENIX Security Symposium, 2021

122 L.A. Galloway and T. Yunusov, "First contact: New vulnerabilities in contactless payments," in Black Hat Europe 2019

123 Ben Nassi, Yisroel Mirsky, Dudi Nassi, Raz Ben-Netanel, Oleg Drokin, Yuval Elovici, Phantom of the ADAS: Securing Advanced Driver-Assistance Systems from Split-Second Phantom Attacks, CCS '20: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security October 2020

124 Wouter Castryck and Thomas Decru1, "An efficient key recovery attack on SIDH", imec-COSIC, KU Leuven, Belgium

Spectre et Meltdown: insécurité des processeurs



Attaque Covert Channel liées aux Micro Architectures

Publiées en 2017, Spectre[125] et Meltdown[126] sont deux attaques qui exploitent des canaux cachés de cache (*Cache side-channel*).

Un cache mémoire stocke le contenu d'adresses dans une page indexée par une partie du poids faible de l'adresse (LSB); de multiples pages sont associées à un jeu de poids fort (MSB) différents. Le principe du *cache side channel* est la mesure de temps de lecture variable selon que la donnée soit présente dans le cache ou pas.

La technique dite "*Evict+Reload*" consiste à remplir le cache avant l'exécution du programme de la victime à l'aide d'instructions de lecture mémoire.

La technique dite "*Flush+Reload*", utilise une instruction spécifique pour vider le cache, avant l'insertion d'un nouveau contenu.

Un "*covert channel*" (canal caché) désigne tout procédé de communication permettant le transfert d'information de manière illicite. un "*cache side channel*" est donc un *covert channel* basé sur des temps de réponse.

Les processeurs utilisent généralement une technique d'exécution d'instructions dite "out of order", selon un algorithme introduit R. M. Tomasulo[127] en 1967. Plusieurs unités exécutent les instructions dont les résultats sont par la suite écrits dans une mémoire cache.

L'exécution spéculative (*Speculative Execution*) est une technique utilisée par les processeurs performants qui consiste à prédire un chemin d'exécution probable. Le résultat d'une instruction de test est évalué, et en cas d'échec de cette prédiction, le

125 KOCHER, P., GENKIN, D., GRUSS, D., HAAS, W., HAMBURG, M., LIPP, M., MANGARD, S., PRESCHER, T., SCHWARZ, M., AND YAROM, Y. "Spectre Attacks: Exploiting Speculative Execution", 2017

126 LIPP, M., SCHWARZ, M., GRUSS, D., PRESCHER, T., HAAS,W., MANGARD, S., KOCHER, P., GENKIN, D., YAROM, Y.,AND HAMBURG, M. "Meltdown" 2018

127 Tomasulo, R. M. An efficient algorithm for exploiting multiple arithmetic units. IBM Journal of research and Development 11, 1 (1967), 25–33.

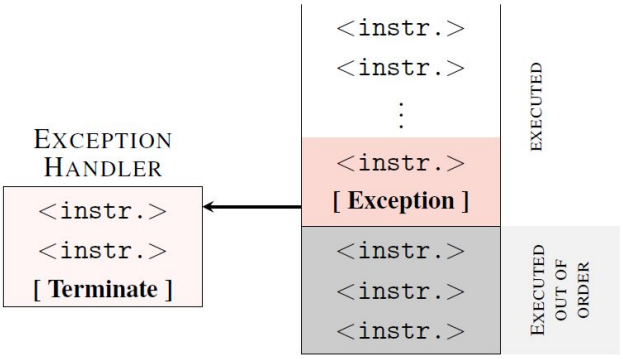
processeur rétablit le contexte antérieur correct, ce qui ralentit le temps de traitement. Cependant la mémoire cache peut conserver la trace de lecture mémoire.

Spectre et Meltdown sont deux procédés d'attaque mettant à profit les techniques *Out of Order Execution* et *Speculative Execution*.

Spectre utilise une variable d'attaque x, l'adresse de chargement d'un élément du tableau array2, dépend de la valeur array1[x]. Si x est supérieure au nombre d'éléments du tableau array1, le contenu de l'adresse mémoire @array2 + (256+array1[x]) est transféré dans le cache. L'instruction spéculative se comporte comme une sonde permettant d'obtenir de contenu de @array1+x

```
if (x < array1_size)
y = array2[array1[x] * 256];
```

```
1 raise_exception();
2 // the line below is never reached
3 access(probe_array[data * 4096]);
```



Escalade de privilège dans Meltdown

Meltdown met à profit un bug Intel relatif à une escalade de privilège. L'exécution spéculative d'une exception permet de basculer du mode user vers le mode superviseur. Le mode superviseur accède à toute la mémoire du système, une sonde permet de lire une adresse liée à la valeur d'une donnée par exemple @probe_array + data*4096

Attaques sur la sécurité du Wi-Fi

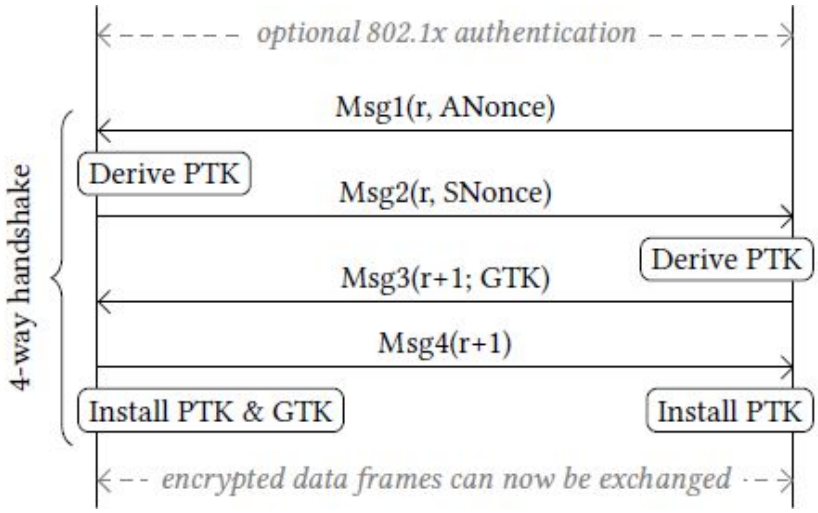
La norme 802.11 (1999) définit un protocole de sécurité radio, le WEP . Son principe consiste à chiffrer les trames à l'aide de l'algorithme RC4 et d'une clé, obtenue par la concaténation d'un secret partagé (de 40 ou 104 bits) et d'un index transporté en clair dans chaque paquet (un champ de 24 bits, noté IV).

RC4 réalise le chiffrement des données en mode flux octets (stream cipher), à partir d'une clé de longueur comprise entre 8 et 2048 bits il génère (à l'aide d'un Pseudo Random Number Generator PRNG) une suite d'octets pseudo aléatoire nommée KeyStream. Cette série d'octets (Ksi) est utilisée pour chiffrer un message en

clair (Mi) à l'aide d'un classique protocole de Vernam, réalisant un ou exclusif (xor) entre Ksi et Mi (Ci = Ksi xor Mi).

L'intégrité des trames est assurée par le chiffrement du CRC. Cette fonction étant linéaire par rapport à l'opération ou exclusif il est possible de modifier un bit dans une trame chiffrée tout en recalculant une valeur correcte du CRC, c'est la technique d'attaque dite *bit flipping*.

L'attaque démontrée par Fluhrer et al. en 2001, permet de recouvrer un secret partagé de 104 bits après l'émission d'approximativement quatre millions de trames chiffrées. Elle utilise des valeurs IV dites résolvantes, de la forme (3+B,255,N) avec B un octet du secret partagé et N une valeur quelconque comprise entre 0 et 255. Environ 60 valeurs résolvantes suffisent à retrouver un octet du secret partagé. Un rapide calcul montre que l'on obtient une valeur résolvante toutes les 216 trames, soit 60 occurrences après environ 4 millions de paquets.



IEEE 802.11 Four Ways Handshake

Un article publié en 2017[128] décrit un scénario d'attaque ("Attaque par réinstallation de clé") du standard IEEE 802.11i (couramment dénommé WPA2) publié en 2004. Malgré une communauté d'utilisateur massive du Wi-Fi, cette faille de sécurité est donc passée inaperçue pendant 14 ans...

Après une procédure d'authentification EAP un secret partagé PMK est calculé par le nœud et le point d'accès. Au terme d'un protocole a quatre passes (*Four Ways Handshake*) les deux parties mettent en place une connexion sécurisée, l'échange des trames est protégée par le protocole CCMP utilisant une clé AES TK (Temporal Key) et un aléa (le CCMP IV).

Les deux premiers paquets du 4 ways handshake comportent deux nombres aléatoires (ANonce et SNonce) et un nombre aléatoire r d'anti rejeu.

[128] Mathy Vanhoef, Frank Piessens, " Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2", CCS'17, October 30-November 3, 2017, Dallas, TX, USA

Une clé PTK (256 bits) est déduite des deux premiers messages selon la relation :

PTK = PRF(PMK, "Pairwise key expansion", Min(AA,SA) || Max(AA,SA) || Min(ANonce, SNonce) || Max(ANonce, SNonce))

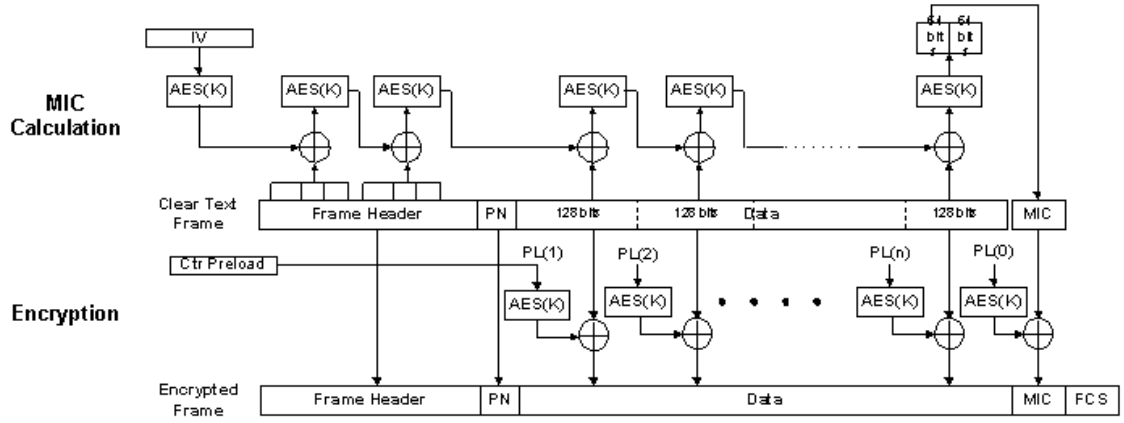
AA = Authenticator Address

SA = Station Address

PRF= Pseudo Random Function

La clé de trame (TK) est le poids faible (128 bits) de PTK.

Les troisième et quatrième paquets du *4 ways handshake* comportent le nombre r+1 (utilisé comme authentifiant) et notifient l'installation (activation) de la clé TK par le point d'accès et le nœud. Par la suite chaque trame MAC chiffrée comporte un Packet Number (PN, 6 octets, initialisé à 1 et incrémenté à chaque paquet).



Le paramètre IV (13 octets) de l'algorithme CCMP est déduit de PN et de l'adresse MAC source). Il comprend 1 octets de priorité, l'adresse MAC (6 octets) de source, et le paramètre PN (6 octets)

Le paramètre compteur (16 octets) est de la forme :

Flag (1octet) || IV || ct (>=0, 2 octets)

CCMP utilise le mode AES-CTR, dans lequel une succession de compteurs croissants sont chiffrés; l'information en clair est protégée à l'aide d'un ou exclusif avec les blocs (16 octets) produits par l'algorithme AES.

Les messages 3 et 4 peuvent être perdus puisqu'ils sont transportés par des trames MAC. Le rejeu du paquet 3 est donc autorisé, ce qui entraine la réinitialisation des paramètres de chiffrements des trames (PN est remis à un). En conséquence on observe un rejeu des blocs (Ksi) de chiffrement produits par l'algorithme l'AES-CTR.

Par exemple si le contenu précédent en clair d'un bloc (Cio = Ksi + Mio) du paquet est connu, on retrouve le contenu en clair (Mi) de la trame chiffrée (Ci= Mi + Ksi) après réinstallation grâce à la relation :

$$Mio + (Ksi + Mio) + (Ksi + Mi) = Mi$$

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[129] 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).

[129] "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				
$\mathtt{Compr}(IV_1,M_1)$	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				
$\mathtt{Compr}(IV_2,M_2)$	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[130] "*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*).

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,…).

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

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[131] chaque année un rapport sur dix les risques de sécurité WEB les plus critiques:

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.

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

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[132], Marsh Ray et Steve Dispensa découvrent une faille dans le protocole SSL/TLS qu'ils nomment *Authentication Gap.*

Cette faiblesse provient du clivage logiciel des protocoles HTTP et TLS. Lors d'une requête HTTPS ordinaire (sans certificat client), le serveur déchiffre l'entête HTTP et découvre que le fichier requis exige un certificat client. Il initialise alors une deuxième session TLS avec mutuelle authentification. L'attaquant se place en MIM (*Man In the Middle*) et détourne la première requête HTTPS sans certificat. Il injecte alors sa propre requête HTTP (codée de telle manière à annuler celle qui sera produite ultérieurement par le client légitime. Par la suite il se comporte en un relais passif transportant la deuxième session TLS, avec le certificat client.

L'attaque BEAST (2011)

En septembre 2011 Juliano Rizzo and Thai Duong ont publié lors de la conférence *ekoparty* à Buenos Aires une attaque dénommée "*Browser Exploit Against SSL/TLS*" (*BEAST*). Cette dernière permet de retrouver par comparaison (Oracle) le contenu chiffré d'un entête HTTP, comme par exemple un cookie (utilisé par une session Paypal dans leur article). L'attaque s'appuie sur une propriété d'un chiffrement en mode CBC (Cipher Block Chaining), lorsque la première valeur (IV_0) est connue; elle a été analysée[133] en 2006 par Gregory V. Bard.

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

---

132 Marsh Ray et Steeve Dispensa, phonephactor.com, Renegotiating TLS, nov. 2009, http://www.phonefactor.com/sslgapdocs/Renegotiating_TLS.pdf

133 Gregory V. Bard "Challenging but feasible Block wise-Adaptive Chosen-Plaintext Attack On SSL", 2006

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).

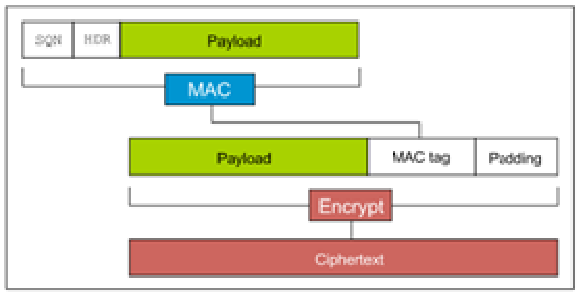


Figure 1: D(TLS) encryption process

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)

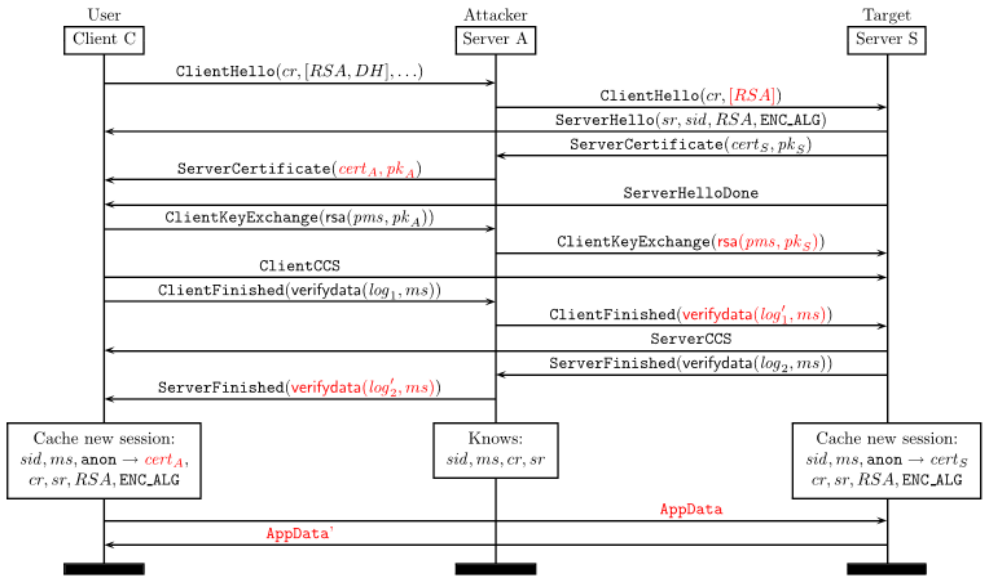
Dans tous les cas de calcul HMAC détecte une intégrité erronée, et un message d'erreur est généré. Le cas d'un padding de longueur supérieur à un, implique un temps calcul plus court.

Attaque TLS Triple Handshake (2014)

Cette attaque[134] 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

[134] https://secure-resumption.com/

de l'attaque est de forger un pre-mastersecret (PMS) identique pour un serveur MIM malveillant et un serveur licite lors d'une session TLS full. Par la suite le serveur MIM est capable de déchiffrer les messages des sessions en mode resume



HeartBleed (2014)

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

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.

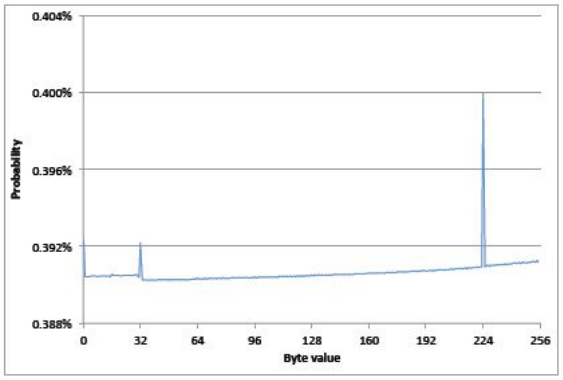
```
struct {
    HeartbeatMessageType type;
    uint16 payload_length;
    opaque payload[HeartbeatMessage.payload_length];
    opaque padding[padding_length];
} HeartbeatMessage;
```

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)

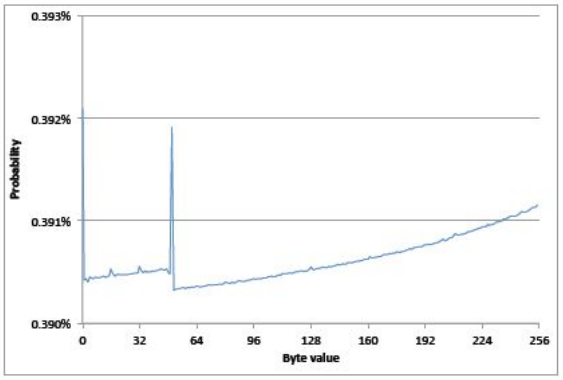
L'attaque RC4 du L'attaque Royal Holloway (2013)

RC4 est l'algorithme le plus utilisé par le protocole TLS. L'attaque *Royal Holloway*, publiée en juillet 2013 par un groupe de chercheurs[135] 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 32ième 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%.

[135] On the Security of RC4 in TLS and WPA", Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, Jacob C. N. Schuldt