

Sécurité (vue rapide)

Introduction

Plusieurs types d'attaques à considérer :

- ▶ attaques locales
 - ▶ accès, ouverture de la machine
 - ▶ exploitation et escalade de privilège depuis un compte utilisateur
- ▶ attaques depuis l'extérieur (réseau)
 - ▶ accès, exploitation, escalade de privilège depuis une connexion réseau
 - ▶ écoute / modification des connexions réseau
 - ▶ DoS
 - ▶ facteur humain
- ▶ attaques passives (sans altération) :
 - ▶ écoute des paquets réseaux, scan de ports...
- ▶ attaques actives (altération) :
 - ▶ DoS, man in the middle, exploit

Accès physique à la machine

Attaque "depuis l'intérieur" : l'attaquant a accès à la machine
(accès régulier en tant qu'utilisateur, vol...)

Solutions :

- ▶ empêcher le boot depuis USB, CD, réseau.
- ▶ mot de passe dans le BIOS et grub
- problème : si on peut ouvrir la machine, on peut effacer le mot de passe du BIOS (enlever la pile)
- ▶ fermer et attacher la machine à clef
(attention, cela se crochète facilement)
- ▶ chiffrer le disque dur (et de préférence, tout le disque dur et le swap)
- ▶ tous les disques durs de portables devraient être chiffrés !

Exploitation de failles

"exploit" : L'attaquant utilise un bug du programme pour lui faire exécuter une fonction qu'il n'aurait pas dû en temps normal

Consequences possibles :

- ▶ plantages
- ▶ appel d'une fonction dans la libc (ex : "return to libc")
- ▶ accès à un shell au niveau de privilège du processus ("shellcode")
- ▶ ...

"Escalade de privilège"

Exemple : on est utilisateur lambda, et on utilise exploite une faille dans un programme setuid root pour pouvoir lancer une commande/fonction avec les priviléges root

Bugs couramment exploités

Problèmes de mémoire

- ▶ dépassement de tableaux (buffer, stack, heap overflow...)
- ▶ problèmes de pointeurs...

Problèmes d'entrées

- ▶ fonctions non sûres (sprintf...)
- ▶ dépassement d'entiers
 - ▶ attention au négatif !
- ▶ chaîne de formatage

```
int main(int ac, char **av) {  
    printf(av[1]);  
    return 0;  
}
```

- ▶ injection (injection SQL...)

...

Bugs couramment exploités

Prévention (voir cours "Bonnes pratiques, débogage et optimisation") :

- ▶ Bonne pratiques de codage
- ▶ Éviter les fonctions réputées dangereuses
- ▶ Vérifier les codes retours
- ▶ Vérifier les dépassements
- ▶ Compiler avec -Wall
- ▶ valgrind
- ...

Exploitations de failles depuis l'extérieur (Réseau)

L'attaquant se connecte à une application via le réseau, et exploite une faille de l'application

Solutions (partielles) :

- ▶ garder le système à jour (mise à jour de sécurité)
- ▶ limiter les logiciels accessibles (et surtout ceux qui peuvent mener à une escalade de priviléges) à ceux nécessaires
- ▶ filtrer les paquets ("firewall") (Linux : iptables ou autres)
- ▶ mots de passe robustes (pas de mots de passe vide, bidon, défaut, même pour les tests !)...

Détecter :

- ▶ analyse de logs (`/var/log/`)
 - ▶ attention, cela peut être effacé par l'attaquant
- ▶ systèmes de détection d'intrusion...

Attaques sur le réseau :

- ▶ Écoute des messages réseaux
- ▶ Manipulation des messages réseaux
- ▶ Spoofing (usurpation d'une adresse)
- ▶ DoS (Denial of Service)
- ...

Sécurité sur le réseau

Les liens réseaux sont généralement considérés comme non sûrs :

- ▶ Tout le monde peut écouter ce qui passe sur le wifi
- ▶ Il est possible d'écouter ce qui passe sur l'ethernet (même si c'est routé)
- ▶ Un routeur IP peut être compromis, ou espionné
- ▶ ...

Solutions :

- ▶ Chiffrer/Signer les messages
- ▶ Certificats
- ▶ (Généralement tous les protocoles ont une version chiffrée)

Cryptographie : chiffrement

Deux grands types de systèmes cryptographiques :

Cryptographie symétrique :

- ▶ il faut la même clé pour chiffrer et déchiffrer
- ▶ rapide

Cryptographie asymétrique :

- ▶ les messages peuvent être chiffrés par tout le monde, via une clé publique
- ▶ ils ne peuvent être déchiffrés que via une clé privée
- ▶ problèmes mathématiques, plus lent

Souvent, les protocoles utilisent les deux : une phase asymétrique pour donner/échanger une clé privée, puis le reste en symétrique.

Principe de Kerckhoffs

Principe de Kerckhoffs :

- ▶ "La sécurité d'un cryptosystème ne doit reposer que sur le secret de la clé"

Maxime de Shannon :

- ▶ "L'adversaire connaît le système"

Plus un algorithme de cryptographie est publique et connu, plus il sera testé et sûr...

Cryptographie symétrique :

- ▶ Même clef pour crypter et décrypter
- ▶ Généralement, travaille sur des blocs de $b = 32 \cdot k$ bits.
- ▶ La fonction de chiffrement et une bijection de 2^b vers 2^b

Exemples :

- ▶ DES (Data Encryption Standard) :
 - ▶ Blocs de 64 bits, clef de 56 bits
 - ▶ Ancien standard, mais devenu bien trop faible. Ne plus utiliser !

- ▶ AES (Advanced Encryption Standard) :

- ▶ Blocs de taille 128, clefs de taille 128, 192 ou 256 bits

Avantages : rapide (opérations simples), souvent en hardware
Inconvénient : les clefs doivent être partagés sur un canal sécurisé

Cryptographie symétrique : Mode d'opération

Si on chiffre une suite de blocs m_0, m_1, \dots , on n'utilise généralement pas la fonction telle quelle sur chaque bloc.

Sinon :

- ▶ Deux blocs identiques seront encodés de la même manière
- ▶ On peut facilement dupliquer et supprimer des bouts de messages...

"Mode d'opération" sur les blocs :

▶ Cipher Block Chaining (CBC) :

$$\Delta c_0 = f_e(m_0 \oplus IV)$$

$$\Delta c_i = f_e(m_i \oplus c_{i-1})$$

▶ Cipher Feedback (CFB) (tolérant aux erreurs de transmission)

$$\Delta c_0 = m_0 \oplus f_e(IV)$$

$$\Delta c_i = m_i \oplus f_e(c_{i-1})$$

▶ ...

(IV : initialisation vector)

Cryptographie asymétrique

Les messages sont chiffrés via une clef publique, et déchiffrés via une clef privée

Exemples :

▶ RSA, El-Gamal, ECC, DH

Problème :

- ▶ plus lent (opérations compliquées)
- ▶ clefs généralement grandes
- ▶ souvent basés sur des problèmes qu'on suppose difficiles...

Cryptographie asymétrique

Basés sur des problèmes mathématiques difficiles :

- ▶ Décomposition en facteurs premiers (RSA, Rabin...)
- ▶ Logarithme discret : (ElGamal, Diffie-Hellman)
 - ▶ Z_p
 - ▶ Courbes elliptiques (ECC) : clefs plus petites
 - ...

...

"Post-quantique" :

- ▶ plus court vecteur dans un réseau (NTRU)
- ...

...

Exemple : Chiffrement RSA (Rivest-Shamir-Adleman)

- ▶ Alice choisit deux nombres premiers p et q , et un entier e
 - ▶ La clé publique est (pq, e)
 - ▶ La clé privée est (p, q, e)
- ▶ Bob chiffre le message m en $c = m^e \pmod{pq}$
- ▶ Alice déchiffre $m = c^f \pmod{pq}$, où
 $ef = 1 \pmod{\varphi(pq)}$
- ▶ $\varphi(pq) = (p-1)(q-1)$ (Indicatrice d'Euler)
- ▶ $m^{ef} = m^{1+k\varphi(pq)} = m \times (m^{\varphi(pq)})^k = m \pmod{pq}$
- ▶ (souvent) difficile de retrouver p et q à partir de pq

Exemple : Chiffrement RSA (Rivest-Shamir-Adleman)

Problèmes :

- ▶ Si m est petit, ou si $\log(m) \times e < \log(pq)$, on peut facilement retrouver m à partir de c
- ▶ Si l'ensemble des possibilités pour m est petit (ex "OUI" ou "NON"), on peut tout essayer
- ▶ Deux blocs identiques sont codés de la même manière.

Solution : "Padding"

- ▶ Une partie importante du message (au moins 8 octets) sont remplis aléatoirement

Exemple 2 : Échange de clef de Diffie-Hellman

- ▶ Alice et Bob se mettent d'accord (publiquement) sur un groupe G (ex : $(\mathbb{Z}/p\mathbb{Z}, \times)$, p premier), et sur un générateur $g \in G$
- ▶ Alice choisit a et Bob choisit b (secrets)
- ▶ Alice envoie g^a à Bob
- ▶ Bob envoie g^b à Alice
- ▶ Alice calcule la clef $c = (g^b)^a$
- ▶ Bob calcule la clef $c = (g^a)^b$
- ▶ Alice et Bob peuvent se servir de c pour chiffrer avec un système symétrique
 - ▶ (souvent) difficile de retrouver a depuis g^a et g (Logarithme discret)
 - ▶ ECC : encore plus difficile...

Taille des clefs & records

NIST (National Institute of Standards and Technology) (2012) :

Sym.	RSA / DH	ECC
80	1024	160-233
112	2048	224-255
128	3072	256-383
192	7680	384-512
256	15360	512+

Record de clef RSA cassée : 768 bits (2010)

Record de clef ECC cassée : 113 bits (2015)

Fonction de hachage cryptographique

Fonction de hachage :
associe à une donnée de taille arbitraire une image de taille fixe

Généralement, $f : 2^k \rightarrow 2^b$ où k est quelconque, et b est un multiple de la taille des mots machine (64, 128, 256...)

Applications :

- ▶ Généralise la somme de contrôle (vérifier qu'un message/fichier n'a pas été modifié)
- ▶ Table de hachage

Fonction de hachage cryptographique

Fonction de hachage cryptographique :

- ▶ rapide à calculer
- ▶ sens-unique : étant donné y , difficile de trouver x tel que $f(x) = y$
- ▶ résistante faible aux collisions : étant donné x , difficile de trouver $x' \neq x$ tel que $f(x) = f(x')$
- ▶ résistante forte aux collisions : difficile de trouver $x \neq x'$ tels que $f(x) = f(x')$

Exemples :

- ▶ MD5 : 128 bits
- ▶ SHA-1 : 160 bits, SHA-256 : 256 bits

Application :

- ▶ Sommes de contrôles (vérifier qu'un message/fichier n'a pas été modifié volontairement) (md5, sha1...)
- ▶ Signatures

Signature numérique

Permet au destinataire d'un message :

- ▶ d'identifier l'émetteur
- ▶ de vérifier que le message n'a pas été modifié

Même principe que la cryptographie asymétrique

Généralement, utilisé conjointement avec une fonction de hachage :

- ▶ On signe le haché du message.

Signature numérique

Exemple : signature RSA :

- ▶ $\text{enc}(m) = m^e$ et $\text{dec}(c) = c^f$ modulo pq
avec $ef = 1 \pmod{(p-1)(q-1)}$
- ▶ enc et dec sont commutatives :
 $\text{enc}(\text{dec}(x)) = \text{dec}(\text{enc}(x)) = x.$
- ▶ Bob envoie un message m à Alice
- ▶ Il calcule le haché $h(m)$ de m
- ▶ Bob envoie m concaténé à $s = \text{dec}(h(m))$
- ▶ Alice vérifie si $\text{enc}(s) = h(m)$

Chiffrement + Signature RSA :

Alice et Bob ont chacun leur cléf

- ▶ Bob envoie un message m à Alice
- ▶ $s = \text{enc}_b(h(m)).$
- ▶ Bob envoie $m' = \text{enc}_a(m|s))$
- ▶ Alice déchiffre $m' : m|s = \text{dec}_a(m')$
- ▶ Alice vérifie si $\text{dec}_b(s) = h(m)$

"Man in the middle"

Alice et Bob doivent préalablement échanger leur clef publiques.

Si cela se fait sur un canal non sécurisé : un attaquant peut modifier tous les messages entre Alice et Bob, il peut substituer les clefs publiques (dont il ne connaît pas clef privées) par des nouvelles clefs publiques qu'il a générée.

Attaque "Man in the middle".

"Man in the middle"

- ▶ Bob envoie sa clef publique pub_b à Alice
- ▶ Oscar intercepte le message, et remplace pub_b par $pub_{b'}$, avant de le renvoyer à Alice
- ▶ Alice pense que la clef de Bob est $pub_{b'}$, et lui envoie un message en chiffrant avec $pub_{b'}$
- ▶ Oscar intercepte le message, le déchiffre avec la clef $priv_{b'}$ et envoie à Bob le message chiffré avec pub_b
- ▶ Bob reçoit un message (éventuellement signé par Alice), et ne s'aperçoit de rien.

"Man in the middle"

Solutions :

- ▶ Échanger les clefs par un canal sûr (par exemple en personne)
- ▶ Certifications de clefs et autorités de confiances :
 - ▶ faire certifier sa clef par une autorité de confiance. La certification réside dans la signature par l'autorité de confiance

Exemple :

- ▶ Certificat : $cert = \text{"Michael Rao, 857736C8"}$
- ▶ Certification par l'autorité CA : $cert | sigCA(hash(cert))$
(après vérification de l'identité)
- ▶ On accepte tous les certificats de CA.

En pratique...

- ▶ Des fonctions de cryptographie sont cassées ou trop faibles : MD4, DES, WEP...
- ▶ Souvent, les problèmes ne viennent pas des fonctions cryptographie, mais de protocoles mal faits
- ▶ Généralement, évitez de designer vous même vos fonctions/protocoles de chiffrement
- ▶ Utilisez si possibles des librairies/protocoles existants et éprouvés !

SSL/TLS

SSL = Secure Sockets Layer

TLS = Transport Layer Security

Protocoles de sécurisation des échanges. ("Couche supplémentaire" dans le modèle par couche)

Permet de :

- ▶ chiffrer
- ▶ authentifier le client et le serveur
- ▶ vérifier l'intégrité des données

Exemple : HTTPS = HTTP sur SSL/TLS,

Programmes / Librairies

- ▶ Librairies implémentant les fonctions cryptographiques courantes : openssl : (implémente SSL et les fonctions cryptographiques bas niveau), libgcrypt...
- ▶ PGP/GPG pour chiffrer/signer les mails.
- ▶ SSH : session, copie de fichiers, système de fichier réseau...
- ▶ Chiffrer les disques : LUKS, encfs (user-space)