Méthodes de cassage des mots de passe...

Denis Ducamp

30 mai 2006

TOULOUSE

Buropolis A - Bâtiment A

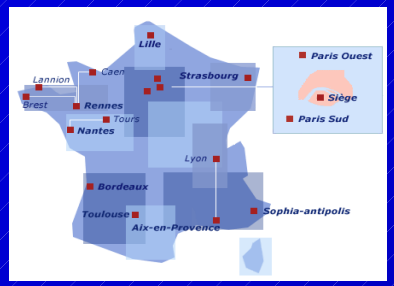
150, rue Nicolas Vauquelin

31100 TOULOUSE

Tél. : 05 34 61 59 59

Fax : 05 34 61 59 58

SII France :
Société de Conseil et d'Ingénierie
Création en 1979
10 agences et 5 bureaux
1.800 personnes (30/03/06)



SII Toulouse

Création en 2000
Effectif : 250 personnes
Métiers :

- **Conception et développement de systèmes embarqués**
- **Réseaux et sécurité informatique**
- **Ingénierie logicielle**

La sécurité chez SII

Les Missions :

- Définition d'architectures sécurisées
- Assistance à maîtrise d'ouvrage
- Expertise en sécurité des S.I. embarqués

…

Le Département Technique Sécurité SII:

- Cellule d'experts en sécurité > capitalisation du savoir, formation, support
- Laboratoire sécurité > recherche et de veille technologique

Partenaires

AIRBUS, THALES AVIONICS, Rockwell Collins, ALCATEL, SIEMENS, AIRBUS CIMPA, freescale semiconductor, MOTOROLA, CLS, Pierre Fabre, UNIVERSITE PAUL SABATIER TOULOUSE III, france telecom, AIR FRANCE

Introduction

- Les mots de passe sont souvent la seule barrière entre un système et son attaquant
  - De leur force dépend la sécurité du système
- Leur force dépend trop souvent du bon vouloir des utilisateurs
  - Quand un système comporte plusieurs milliers d'utilisateurs il doit donc exister plusieurs façons évidentes d'accéder au système
- Les mots de passe sont protégés sur un système
  - « chiffrement », droits d'accès, politique
- Ainsi que lors de leurs transits sur un réseau
  - « chiffrement »

Sommaire

- Les « chiffrements » systèmes
- Algorithmes systèmes courants
  - Unix (avec graines)
  - Windows (sans graine)
- Les attaques
  - Dictionnaires
  - Force brute (intelligente)
  - Dictionnaires pré-calculés
  - Tables
- Performances
- Durcissement des mots de passe

Plan :
les « chiffrements » systèmes

- Hachage
- Graines
- Vérification d'un mot de passe
- Fichiers de mots de passe

Hachage

- Le hachage est une fonction non réversible permettant d'obtenir une image d'un mot de passe.
  - Initialement basée sur une fonction de chiffrement
- Le mot de passe est transformé en clé pour chiffrer une chaîne connue
  - Une fonction de hachage classique peut être utilisée
- La fonction principale est souvent appelée de façon itérative
  - Éventuellement avec d'autres fonctions (translation, rotation, xor...) pour complexifier la cryptanalyse
- La probabilité d'une collision est exceptionnelle

Graines

- Permettent que le même mot de passe donne des images différentes
- Généralement constituée de données aléatoires

- Si le nombre de graines différentes utilisées est suffisant
  - Ralenti le cassage hors-ligne
    - Chaque empreinte doit être testée individuellement
  - Empêche les attaques par dictionnaires pré-calculés

Vérification d'un mot de passe

- L'utilisateur entre son mot de passe
- Le système récupère la graine utilisée pour stocker le mot de passe
- Le mot de passe est « chiffré » en utilisant cette graine
- Si le résultat obtenu est l'empreinte stockée alors on considère que l'utilisateur a entré le bon mot de passe
  - La probabilité d'un faux positif dû à une collision devant être exceptionnelle

Fichiers de mots de passe

- Historiquement sous Unix le fichier des utilisateurs (*/etc/passwd*) hébergeait les mots de passe
- Actuellement il existe un fichier de mots de passe
  - */etc/shadow*, */etc/master.pwd*, etc suivant les systèmes
  - Celui-ci n'est accessible qu'à l'administrateur

- Sous Windows ces données sont dans la base de registre : fichier *SAM*.
  - Ce fichier est verrouillé par le système au démarrage
  - Les clés ne sont accessibles qu'au compte *SYSTEM*

Plan : algorithmes systèmes courants

- Unix
  - DES
  - MD5
  - Blowfish
- Windows
  - Lanman (LM)
  - NTLM

DES

- La première empreinte sous Unix
- Basée sur une version modifiée de DES itérée 25 fois
  - Dépendante d'une graine de 12 bits (4096 possibilités)
- Le mot de passe est limité à 8 caractères de 7 bits
  - L'espace des mots de passe est donc relativement faible
- Possibilité de mise au point de dictionnaires pré-calculés
- Avec un très grand nombre de comptes
  - Les collisions de graines seront certaines
  - Permettant ainsi d'accélérer le cassage hors ligne
- DES est « mort » alias « 56 bits en 56 heures » :

http://www.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/

MD5

- Empreinte la plus courante aujourd'hui sous Unix
- Basée sur une itération de la fonction md5 (1000x)
  - Avec d'autres fonctions simples
  - Et dépendante d'une graine de 6 à 48 bits
- La longueur du mot de passe est en théorie illimitée

Blowfish

- Dernière des empreintes Unix créée pour OpenBSD
- Basée sur une itération de la fonction blowfish
  - Avec d'autres fonctions simples
  - Et dépendante d'une graine de 128 bits
- Le nombre d'itérations est paramétrable
  - De 2^4 à 2^31
  - À multiplier par deux tous les un an et demi (loi de Moore)
- La longueur des mots de passe est limitée à 55 caractères

LanMan (LM)

- Empreinte créée par IBM et utilisée par Microsoft
- Le mot de passe est passé en majuscules
  - Puis divisé en deux mots de passe de 7 caractères
  - Chacun donnant une clé de chiffrement DES
- Une chaîne connue est chiffrée en DES avec chaque moitié
- La concaténation de ces deux chiffrés donne l'empreinte LanMan

NTLM

- Empreinte créée par Microsoft pour Windows NT
- Le mot de passe est codé en Unicode
- Le tout est haché en md4 pour donner l'image NTLM

- Les deux empreintes, LanMan et NTLM, sont enregistrées dans la SAM
  - Et obfusquées par chiffrement avec le RID de l'utilisateur
  - Mais avec les mêmes codons RC4 sous NT4preSP6 !!!
- Il est conseillé de désactiver l'enregistrement de l'empreinte LanMan
  - Mais ce n'est effectif pour chaque utilisateur que lors de son prochain changement de mot de passe

Plan: les attaques

- En ligne vs. Hors ligne
- Dictionnaires
- Force brute
  - Force brute intelligente
- Dictionnaires pré-calculés
  - Tables de Hellman
  - Tables *rainbow*

En ligne vs. Hors ligne

- Le cassage en ligne s'effectue via des tests réseaux ou locaux
- Celui-ci est généralement lent car
  - Les fonctions systèmes ne sont pas forcément optimisées
  - Souvent des *timeouts* sont introduits
- Et peu discret
  - Chaque échec génère un événement qui doit être journalisé
  - Peut générer des verrouillages de comptes
- Le cassage hors ligne s'effectue après un accès à la base des mots de passe
  - Pas forcément à cause du piratage du serveur visé
- Celui-ci est généralement rapide et discret

Dictionnaires

- Tous les mots de divers dictionnaires sont testés
  - Sans et avec transformations
    - Tout en minuscules
    - Tout en majuscules
    - Seule l'initiale en majuscule
    - À l'envers
    - Préfixe/suffixe avec un caractères « autre »
    - Changement de lettres par des chiffres ressemblant : « h4ck3r »
  - Les dictionnaires peuvent être différents :
    - Prénoms et noms de famille
    - Noms de personnes / personnages (acteurs... / SF, BD, ciné...)
    - Linguistiques (français, anglais, allemand, espagnol, *klingon*...)
    - Dictionnaires spécialisés (chimie...)

Force brute et force brute intelligente

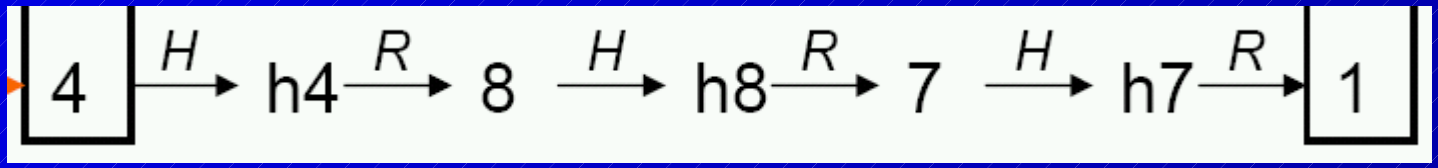
- Sont testées toutes les combinaisons possibles pour un ensemble donné de caractères
- Méthode généralement lente
  - Sauf si le nombre de caractères différents et la longueur maximale sont limités.
    - Par exemple LanMan est faisable en quelques mois
- La force brute intelligente permet de tester toutes les combinaisons possibles
  - mais dans un ordre permettant de tester d'abord les mots de passe les plus probables
- Par exemple *John the ripper* utilise des statistiques longueurs, premiers et derniers caractères, suites de caractères...

Dictionnaires pré-calculés

- Des couples (mdp, hash) sont enregistrés dans une base
  - Le cassage se résume à une recherche en base
- Ne fonctionne qu'avec les algorithmes sans graine
  - LanMan et NTLM
  - md2, md4, md5, ripemd160, sha1, MySQL, PIX...
- ou avec peu de graines
  - Oracle : le *login* joue le rôle de graine
    - 2 comptes administrateurs
      - *sys* (*change_on_install*)
      - *system* (*manager*)
  - DES : 4096 possibilités soit 32Ko par mot de passe
    - Qcrack n'enregistre que le premier octet de chaque empreinte
      Soit 4Ko par mot de passe

Dictionnaires pré-calculés
Tables de Hellman

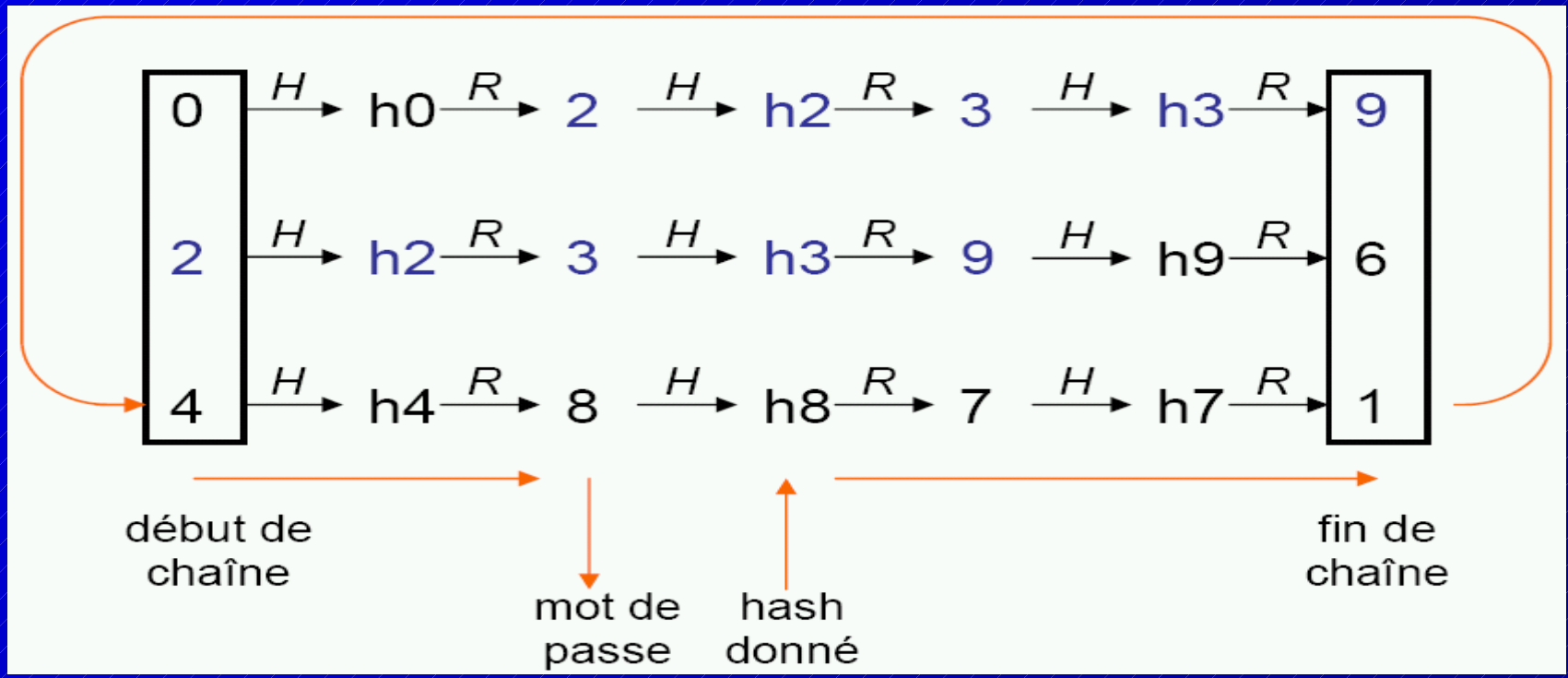
- Des chaînes, de longueur t, sont calculées
  - À partir d'une fonction R : R(hash) -> mdp



    - Les mdp initiaux sont tirés aléatoirement
  - Pour chaque chaîne seuls mdp_1 et $hash_t$ sont enregistrés
    - Ce qui divise par plusieurs milliers la taille totale des tables, soit quelques centaines de Go pour LanMan
- L'efficacité de chaque chaîne complémentaire diminue
  - Plusieurs tables sont calculées avec des fonctions R différentes (en fait R est dépendante du numéro de table)
  - Pour avoir une réussite maximale, l'espace des mots de passe doit être balayé plusieurs fois

Dictionnaires pré-calculés
Tables de Hellman

- Pour casser une empreinte E
  - Pour tout i de 0 à t : calculer $H(R(E))^i$
  - Si un $hash_t == H(R(E))^i$ alors calculer $M = R(H(mdp_1))^{t-i}$



- Mais ce résultat peut être un faux positif
  - Il faut donc vérifier que H(M) == E

Dictionnaires pré-calculés
Optimisations de tables

- Beaucoup de faux positifs sont dus à des fusions de chaînes
  - Les tables parfaites sont constituées en retirant les chaînes qui fusionnent (Borst, Preneel, and Vandewalle 1998)
  - Une telle table a un nombre maximal de 2N / t+2 chaînes
    - Et une probabilité de 86% de réussite
    - Soit 98,7% pour 2 tables, 99,7% pour 3 et 99,96% pour 4
- Le temps de recherche d'un hash dans une table est important.
  - Le dernier hash peut être un point distinctif (Rivest 1982)
    - par ex. les derniers bits sont tous nuls
  - Les chaînes doivent alors être de taille variable
  - Ces points permettent de détecter les boucles et les fusions

Dictionnaires pré-calculés
Tables rainbow

- La fonction R dépend du numéro de colonne (Oechslin 2003)
- Permet de limiter les fusions de chaînes aux seuls cas de collision dans une même colonne
  - Une table rainbow de m.t chaines a la même probabilité que t tables classiques de m chaines.
- Pour chaque table rainbow il est nécessaire de faire $O(t^2)$ calculs au lieu de O(t) pour une table classique
  - Mais le nombre maximum de calculs est pour :
    - 1 table rainbow : 1 x t(t-1)/2
    - t tables classiques : t x t
  - soit un gain de 50%

Dictionnaires pré-calculés
Autres optimisations de tables

- La majorité des calculs lors d'un cassage sont dus à des faux positifs. Il peut être intéressant de détecter de façon précoce des faux positifs
- Des *check-points* sont enregistrés à différents endroits des chaînes (Oechslin 2005)
  - Doivent être faciles à calculer
    - Par exemple le dernier bit du hash
  - Et prendre peu de mémoire
    - Pour être sauvés dans les espaces « vides » des tables
- Lors d'un cassage, les calculs sont arrêtés dès qu'un calcul ne correspond pas au *check-point* correspondant
- En pratique 6% de mémoire supplémentaire permet de diminuer de 30% le temps de cassage

Plan : performances

- John the ripper
- Rainbow crack

John the ripper algorithmes par défaut

- fuji ~ # john -test
- Benchmarking: Traditional DES [64/64 BS MMX]... DONE
  Many salts: 653772 c/s real, 655082 c/s virtual
  Only one salt: 586675 c/s real, 587850 c/s virtual
- Benchmarking: BSDI DES (x725) [64/64 BS MMX]... DONE
  Many salts: 21478 c/s real, 21478 c/s virtual
  Only one salt: 21077 c/s real, 21120 c/s virtual
- Benchmarking: FreeBSD MD5 [32/32]... DONE
  Raw: 4335 c/s real, 4344 c/s virtual
- Benchmarking: OpenBSD Blowfish (x32) [32/32]... DONE
  Raw: 291 c/s real, 291 c/s virtual
- Benchmarking: Kerberos AFS DES [48/64 4K MMX]... DONE
  Short: 232089 c/s real, 232554 c/s virtual
  Long: 577916 c/s real, 580232 c/s virtual
- Benchmarking: NT LM DES [64/64 BS MMX]... DONE
  Raw: 5271K c/s real, 5346K c/s virtual

John the ripper algorithmes supplémentaires

- Benchmarking: Apache MD5 [32/32]... DONE
  Raw: 4339 c/s real, 4339 c/s virtual
- Benchmarking: mysql [mysql]... DONE
  Raw: 1274K c/s real, 1279K c/s virtual
- Benchmarking: Netscape LDAP SHA (SSE2 4x) [SHA1]... DONE
  Raw: 2305K c/s real, 2314K c/s virtual
- Benchmarking: NT MD4 [TridgeMD4]... DONE
  Raw: 1592K c/s real, 1592K c/s virtual
- Benchmarking: Lotus5 [Lotus v5 Proprietary]... DONE
  Raw: 146973 c/s real, 147268 c/s virtual
- Benchmarking: M$ Cache Hash [mscash]... DONE
  Raw: 1024K c/s real, 1024K c/s virtual
- Benchmarking: Raw MD5 (SSE2 4x) [raw-md5 SSE2]... DONE
  Raw: 4357K c/s real, 4375K c/s virtual
- Benchmarking: Eggdrop [blowfish]... DONE
  Raw: 7715 c/s real, 7731 c/s virtual

John the ripper algorithmes supplémentaires

- Benchmarking: Raw SHA1 (SSE2 4x) [raw-sha1 SSE2]... DONE
  Raw: 2387K c/s real, 2392K c/s virtual
- Benchmarking: MS-SQL (SSE2 4x) [ms-sql]... DONE
  Raw: 1733K c/s real, 1733K c/s virtual
- Benchmarking: HMAC MD5 (SSE2 4x) [hmac-md5 SSE2]... DONE
  Raw: 1563K c/s real, 1566K c/s virtual
- Benchmarking: WPA PSK (SSE2 4x) [pwa-psk SSE2]...
  FAILED (cmp_all)
- Benchmarking: Oracle [oracle]... DONE
  Raw: 386874 c/s real, 393152 c/s virtual
- Benchmarking: NT MD4 MMX (SSE2 4x) [bartavelle]... DONE
  Raw: 4704K c/s real, 4723K c/s virtual

- fuji ~ # fgrep "model name" /proc/cpuinfo
  model name : Intel(R) Pentium(R) M processor 1.73GHz
- fuji ~ # cat /etc/gentoo-release
  Gentoo Base System version 1.6.14

John the ripper temps maximaux

- fuji # wc -l all
  3917193 all
- fuji # john -w=all -ru -stdout > /dev/null
  words: 147945837 time: 0:00:02:36 100% w/s: 945522 current: Wysifigging
- fuji # john -w=all -ru passwd
  Loaded 1 password hash (NT LM DES [64/64 BS MMX])
  guesses: 0 time: 0:00:00:51 100% c/s: 516749 trying: ZEEGSIN – ZYMOSIN
- fuji src # john -ext=double passwd
  Loaded 1 password hash (NT LM DES [64/64 BS MMX])
  guesses: 0 time: 0:00:00:00 c/s: 1584K trying: ZZYYZZY - ZZZZZZZ
- fuji src # john -ext=lanman passwd
  Loaded 1 password hash (NT LM DES [64/64 BS MMX])
  guesses: 0 time: 0:00:43:25 c/s: 3206K trying: ZZZZZZE - ZZZZZZZ

Rainbow crack par défaut et supplémentaires

- fuji ~ # rtgen lm alpha 1 7 0 -bench
  lm hash speed: 1096491 / s
  lm step speed: 859106 / s
- fuji ~ # rtgen md5 loweralpha 1 7 0 -bench
  md5 hash speed: 1086956 / s
  md5 step speed: 853242 / s
- fuji ~ # rtgen sha1 loweralpha 1 7 0 -bench
  sha1 hash speed: 939849 / s
  sha1 step speed: 750750 / s

- fuji ~ # rtgen ntlm loweralpha 1 7 0 -bench
  ntlm hash speed: 1091703 / s
  ntlm step speed: 844594 / s
- fuji ~ # rtgen md2 loweralpha 1 7 0 -bench
  md2 hash speed: 124192 / s
  md2 step speed: 120365 / s
- fuji ~ # rtgen md4 loweralpha 1 7 0 -bench
  md4 hash speed: 1126126 / s
  md4 step speed: 868055 / s
- fuji ~ # rtgen ripemd160 loweralpha 1 7 0 -bench
  ripemd160 hash speed: 776397 / s
  ripemd160 step speed: 634517 / s

Rainbow crack temps maximaux

	real	user(1)	dont crypt.	fa(2)	cs(3)	t	%	.gz	Go
Alpha	1m04	32s	21.93s	66,3 %	26	2100	99,90	55.1%	0.27
Alphanum	4m34	1m21	24.01s	38,9 %	36	2400	99,04	49.8%	1.5
Alphanum14	32m02	9m21	181,65s	47,5%	50	5700	99,90	41.7%	11
All (4)	56m01	13m40	145,93s	44,5%	68	9000	~ 80	34.1%	20

(1) dont décompression en temps réel des tables

transfert du disque : 6 à 7Mo/s

CPU lors de la décompression : 20 à 25%

(2) % de pas dus à des fausses alarmes

(3) nombre de caractères différents

toutes les tables prennent en compte les mots de passe de 1 à 7 caractères de long

(4) 2 tables sur 8 seulement.

réussite des 8 tables : 99,90

Faisabilité pour d'autres algorithmes

- NTLM
  - Problèmes : 96+ caractères et longueur « illimitée »
    - Se limiter à des longueurs raisonnables
    - Et à un ensemble de caractères raisonnables
  - Points faibles : pas de durcissement et rapidité de calcul
- DES
  - Problèmes : 4096 graines, 96 caractères
  - Points faibles : longueur limitée à 8
    - Accessible sur des longueurs plus petites
    - Et des ensembles de caractères limités
- MD5, Blowfish...
  - Infaisables car le nombre de graines est trop important

Faisabilité pour d'autres algorithmes

- Tables :
  - Payantes
    - http://sarcaprj.wayreth.eu.org/
    - http://www.rainbowtables.net/
    - etc.
  - Gratuites :
    - http://rainbowtables.shmoo.com/
    - http://rainbowcrack.com/
    - http://wired.s6n.com/files/jathias/
    - etc.
  - Cassage en ligne
    - http://www.plain-text.info/
    - etc.

Plan : durcissement des mots de passe

- Politique de mots de passe
  - Mauvais exemples
  - Stockage des mots de passe
- Vérification de la force

Politique de mots de passe

- Une politique de mots de passe peut imposer :
  - Une fréquence de changement minimale
    - Et un temps minimal entre deux changements
  - Un historique
  - Un verrouillage de compte après un nombre maximum d'erreurs consécutives de mot de passe
    - Avec un temps minimum entre deux erreurs avant ré-initialisation
    - Et un temps de verrouillage après la dernière erreur
  - Une force minimale
    - Une longueur minimale
    - Une complexité minimale
      - Par exemple : des caractères dans un nombre minimal de catégories différentes

Politique de mots de passe mauvais exemples

- Une fréquence de changement trop élevée
  - Incite les utilisateurs à recycler leurs mots de passe
  - Ou à utiliser des mots de passe « cycliques » : printemps, été...
- Un temps minimal trop long entre deux changements
  - Empêche le changement en cas de suspicion de corruption
- Un nombre maximum trop faible d'erreurs consécutives avant verrouillage
  - Ou un temps trop faible entre deux erreurs
  - Ou un temps trop long de verrouillage après la dernière erreur
    - Génèrent de nombreux faux positifs et donc des appels au *help-desk*
- Une force minimale
  sans prise en compte des méthodes de cassage
  ni des spécificités des fonctions de hachage
  est inefficace. Par exemple : *Bonjour1*

Stockage des mots de passe

- Stocker les « mots de passe » de façon sécurisée
  - Dans un fichier accessible seulement aux administrateurs
  - Ou à une application système (ldap)
- Éviter les bases partagées
  - Ou les utiliser de façon sécurisée, i.e. ne pas utiliser le compte d'administration pour récupérer les *credentials*
- Stocker une image non réversible
  - Pas de mot de passe en clair ou chiffré avec une clé secrète
  - Utiliser un maximum de graines différentes
- Utiliser un algorithme aussi costaud que possible
  - Unix : Blowfish si disponible, MD5 sinon
  - Windows : désactiver LanMan (défaut de WinXP SP2)

Vérification de la force

- Lorsque l'utilisateur entre un nouveau mot de passe
- Celui-ci n'est accepté que s'il suit certaines règles
  - Règles basiques sus-citées
  - Simulation de cassage par un logiciel de type crack
    - Par exemple *pam_crack* sous Unix, basé sur *libcrack*
    - Extrêmement rapide puisque les calculs se font sans chiffrement
  - Force d'une passe-phrase
    - Suite de mots séparés par des caractères spéciaux
      - Nombre minimal de mots
      - Longueur minimale de chaque mot
      - Nombre minimal de caractères « autres »
    - L'espace des mots de passe est beaucoup plus grand
      - Mais est beaucoup plus simple à se rappeler qu'un mot de passe aléatoire
    - Par exemple pam_passwdqc sous Unix

Références

- Cassage et durcissement des mots de passe
  - Misc 2 : Windows

    http://www.hsc.fr/ressources/articles/mdp_misc2/index.html.fr
  - Misc 5 : Unix

    http://www.hsc.fr/ressources/articles/mdp_misc5/index.html.fr
- John the Ripper password cracker

  www.openwall.com/john/
- Project RainbowCrack

  www.antsight.com/zsl/rainbowcrack/
- ophcrack et les publications de Philippe Oechslin

  http://lasecwww.epfl.ch/~oechslin/projects/ophcrack/

Remerciements

SII mon actuel employeur

Merci pour le support de la DTS

DTS le groupe de travail sécurité de SII

Merci pour vos relectures et votre support

HSC mon premier employeur

Merci pour tout ce que vous m'avez appris

et de m'avoir fourni tant de mots de passe ;-)

Et un grand merci à vous tous pour votre écoute :-)

Méthodes de cassage des mots de passe...