WAVESTONE

Cassage de mots de passe

Méthodes, outils et mesures de protection

21/02/2017

OSSIR

AGENDA

01 « Cassage de mots de passe »

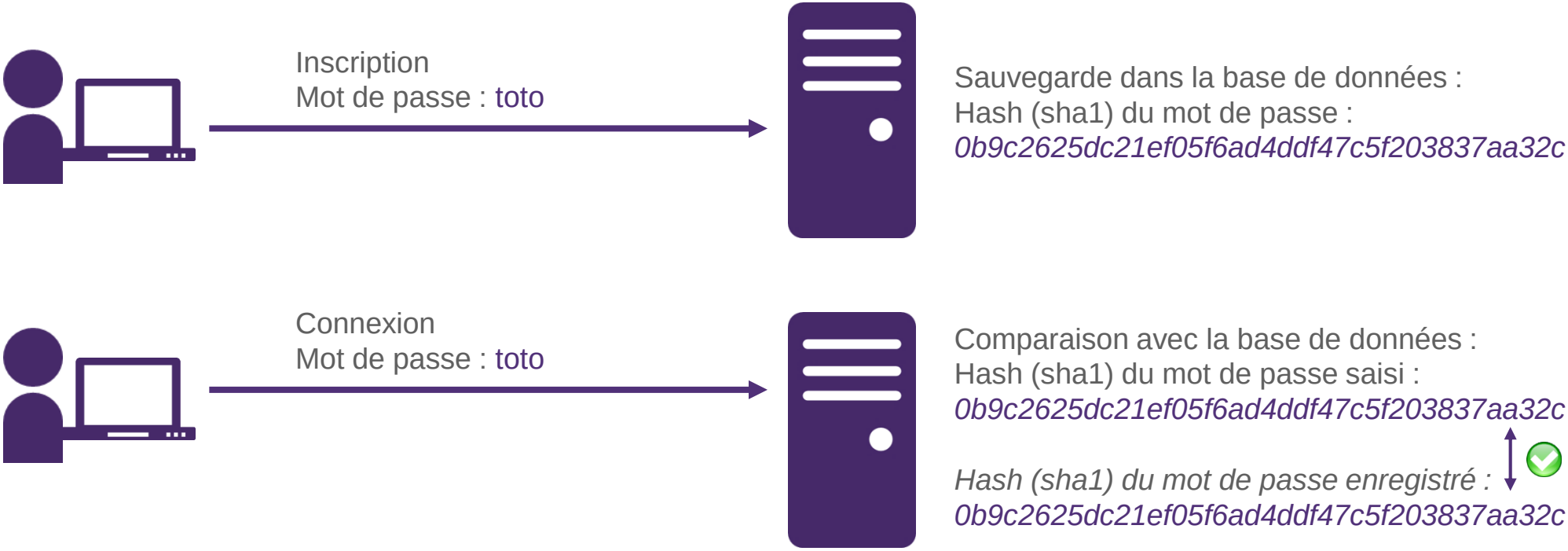
02 Nouvel outil de gestion des cassages

03 Protections contre le cassage de hash

/ 01 « Cassage de mots de passe »

Qu'est-ce qu'un hash de mot de passe ?

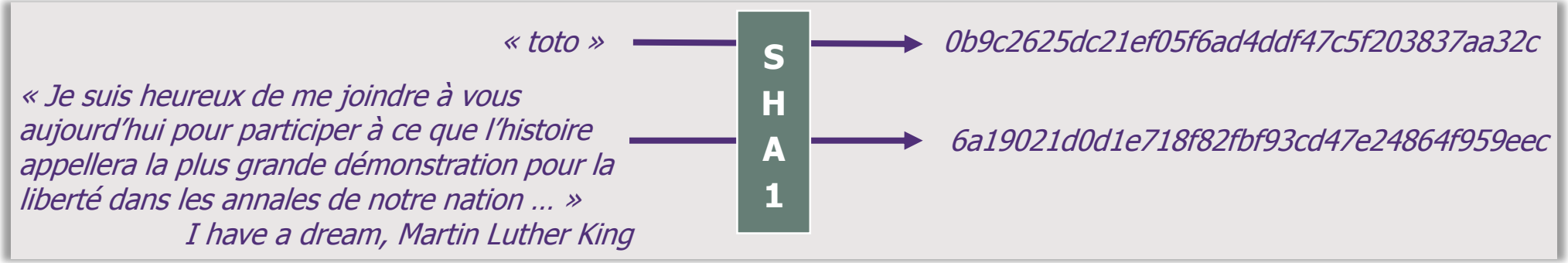
En règle générale, les mots de passe des utilisateurs ne sont pas enregistrés en clair dans les bases de données



Ainsi, en cas de compromission, l'attaquant n'a pas directement accès aux mots de passe utilisateurs en clair

Principe du hachage

Une **fonction de hachage cryptographique** est une fonction qui va retourner pour un message de taille inconnue un hash de **taille fixe**



Et qui respecte les propriétés suivantes :

Résistance à la préimage

Étant donné un hash *H*, il ne doit pas être possible de calculer un message *M* tel que *hash(M) = H*

Résistance à la seconde préimage

Étant donné un message M_1, il ne doit pas être possible de calculer un message M_2 différent de M_1 tel que $hash(M_1) = hash(M_2)$

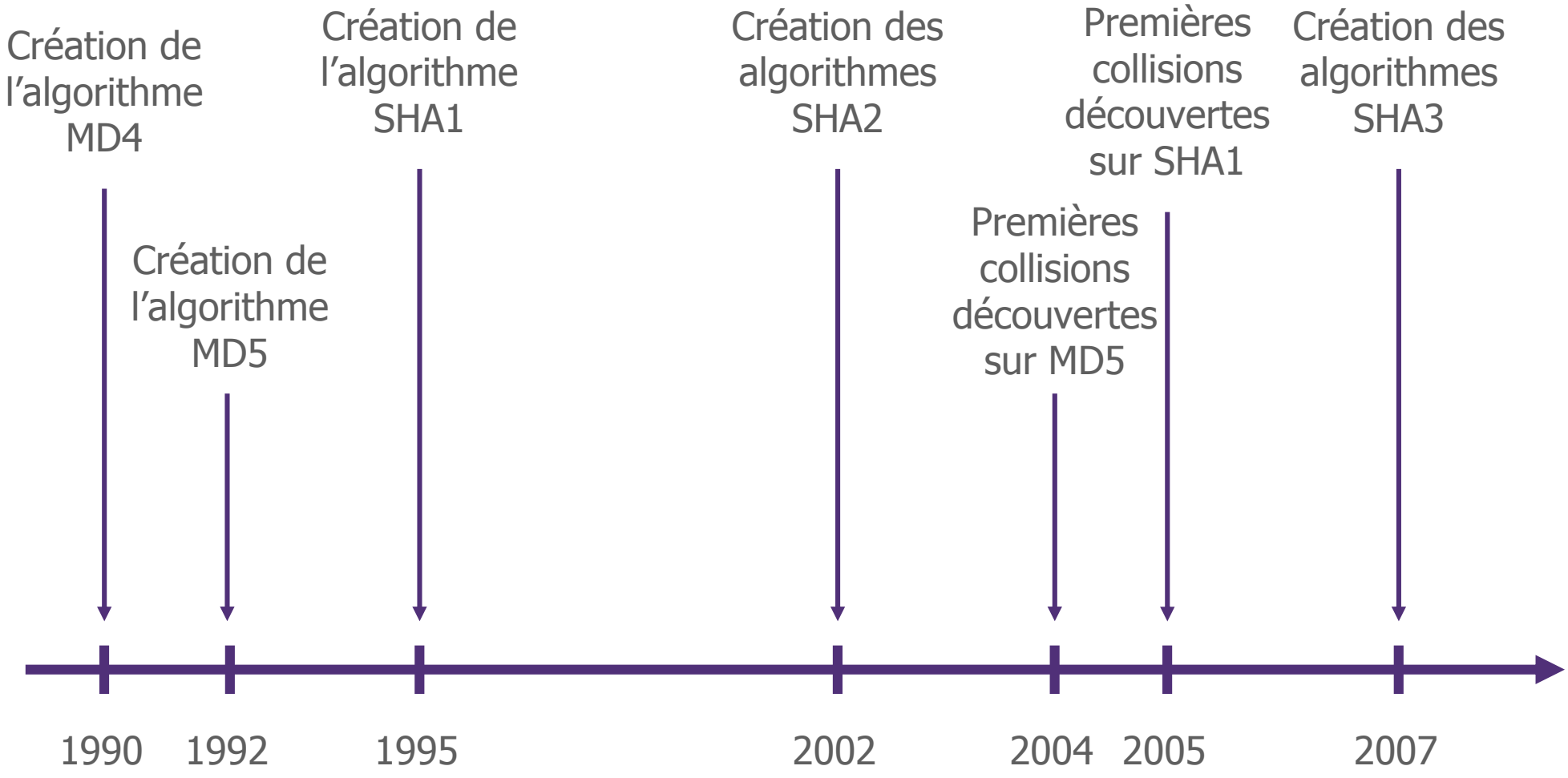
Résistance aux collisions

il ne doit pas être possible de trouver deux message M_1 et M_2 différents tels que $hash(M_1) = hash(M_2)$.

« Effet avalanche »

La modification d'un bit du message résulte d'une probabilité proche de 50% de changement de chacun des bits du hash.

Historique des algorithmes de hachage



Pourquoi casser un hash de mot de passe ?

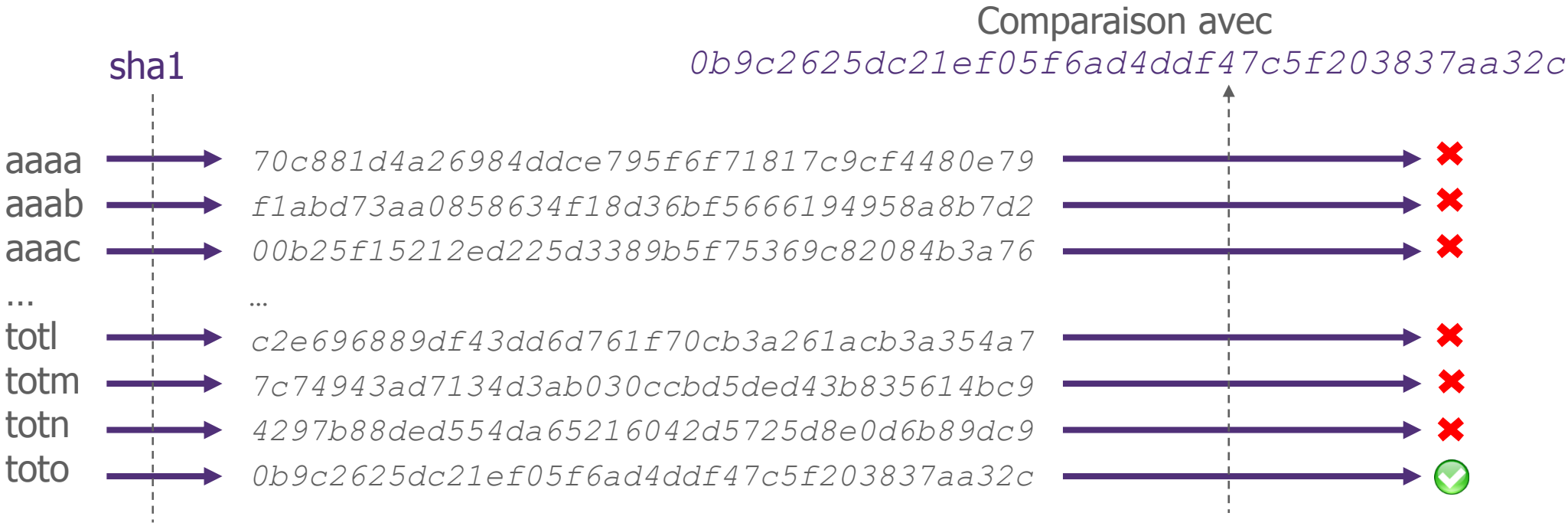
Les fonctions de hachage sont conçues pour être **à sens unique** (être impossibles à remonter), et éviter les collisions (deux entrées différentes produisant deux hash identiques)

Afin d'utiliser un mot de passe lorsqu'on connaît son hash, il est nécessaire de le « casser » pour retrouver le mot de passe original et pouvoir le ré-utiliser *

* : sauf dans les environnements Windows où le hash peut être utilisé à la place du mot de passe (technique Pass-The-Hash)

Comment casser un hash de mot de passe ?

Méthode du bruteforce (« méthode naïve »)



Comment casser un hash de mot de passe ?

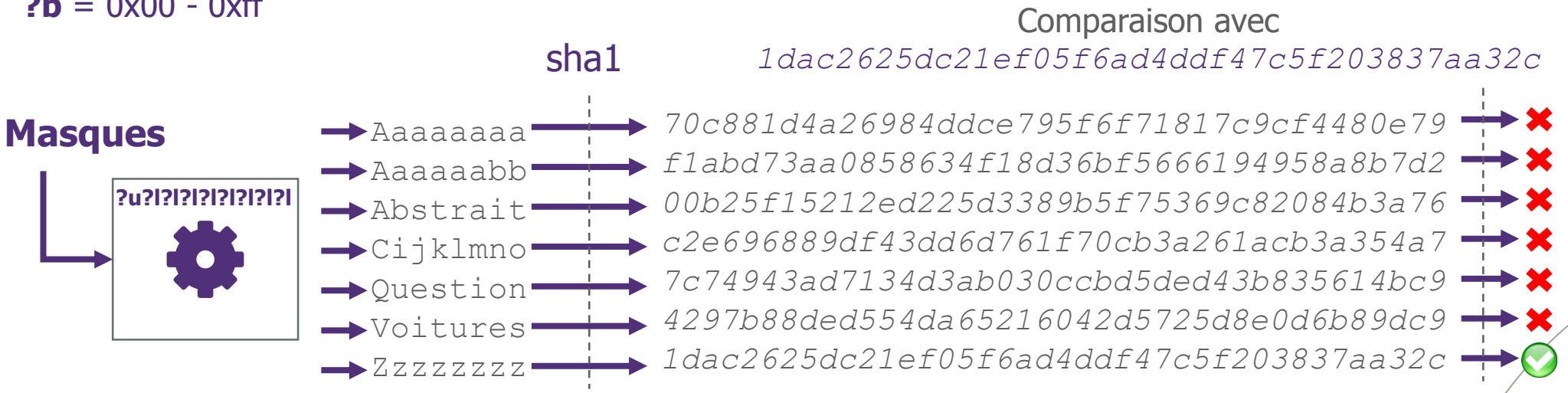
Méthode par masque

Exemples des structures de base existantes pour construire les masques :

?l = abcdefghijklmnopqrstuvwxyz
?u = ABCDEFGHIJKLMNOPQRSTUVWXYZ
?d = 0123456789
?s = «espace»!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
?a = ?l?u?d?s
?b = 0x00 - 0xff

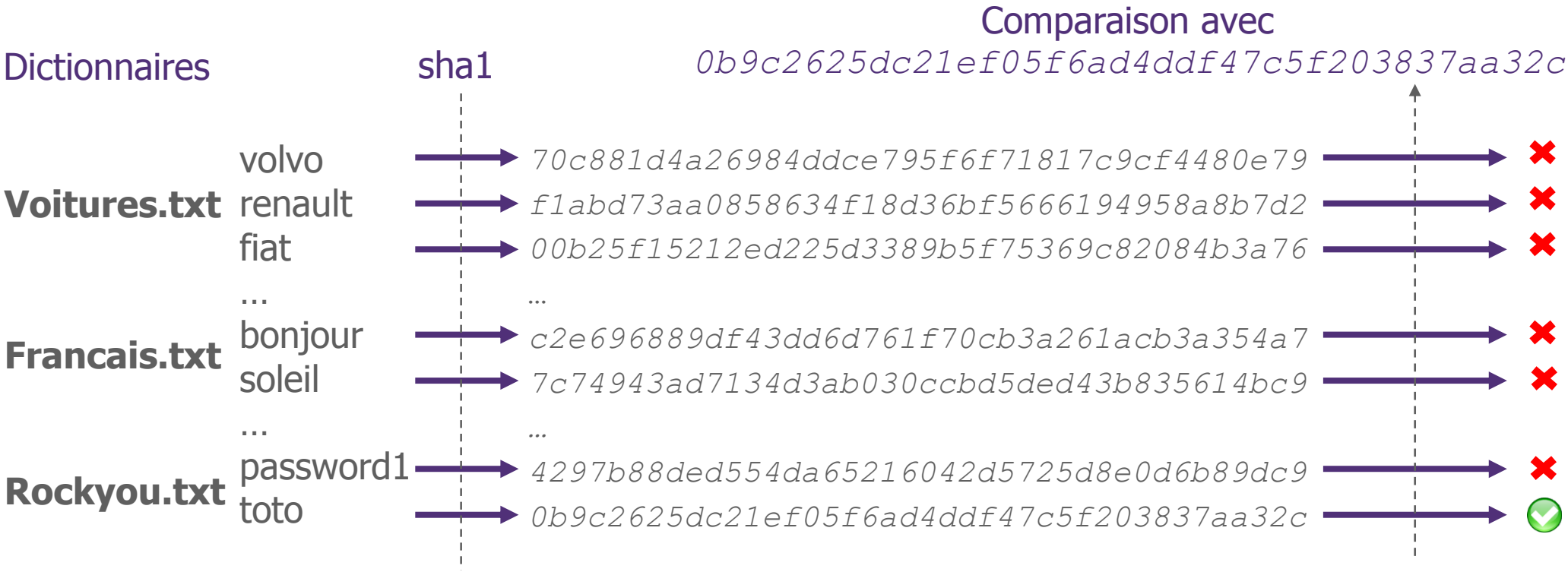
P	a	s	s	w	o	r	d	1	!
?u	?l	?l	?l	?l	?l	?l	?l	?d	?s

Exemple de candidat respectant le masque `?u?l?l?l?l?l?l?l?d?s`



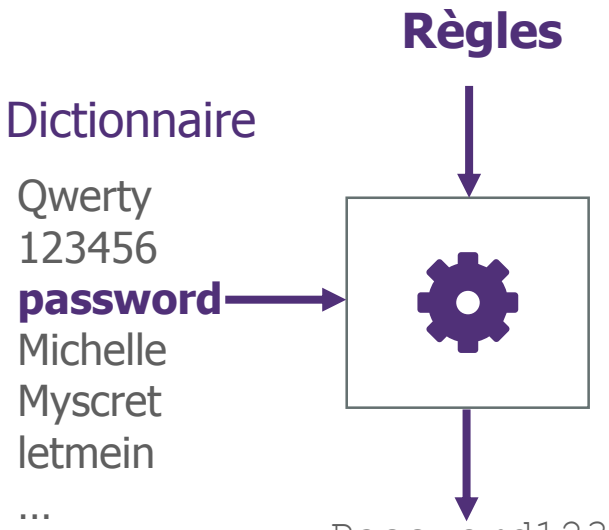
Comment casser un hash de mot de passe ?

Méthode par dictionnaire



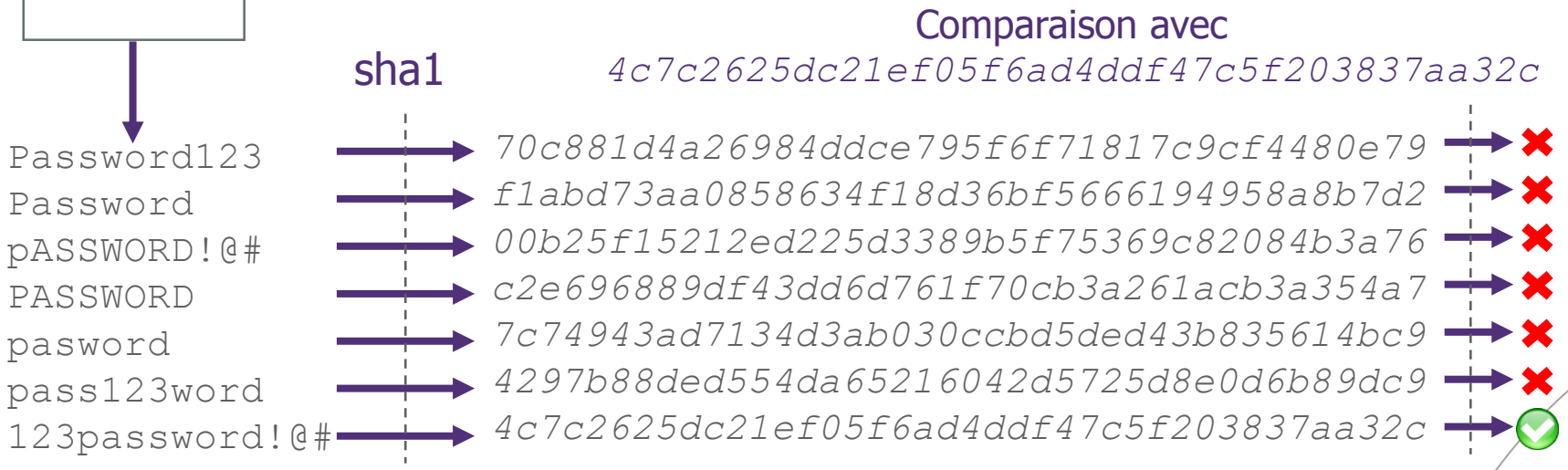
Comment casser un hash de mot de passe ?

Méthode par dictionnaire avec variations



Exemples de règles de variation :

- Remplacer le premier caractère par sa majuscule
- Ajouter des chiffres à la fin
- Ajouter des caractères spéciaux à la fin
- Supprimer une lettre au milieu



Comment casser un hash de mot de passe ?

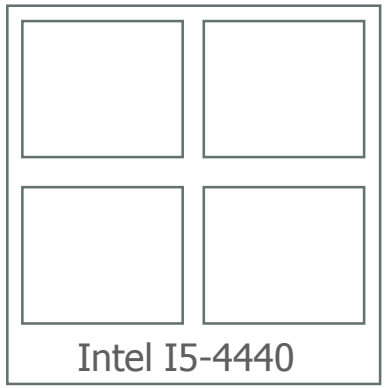
Autres méthodes

D'autre méthodes existent, notamment :

- Brute-force avec chaînes de Markov
- Rainbow tables
- Rainbow tables probabilistes
- Masque et dictionnaire
- Algorithmes génétiques
- …

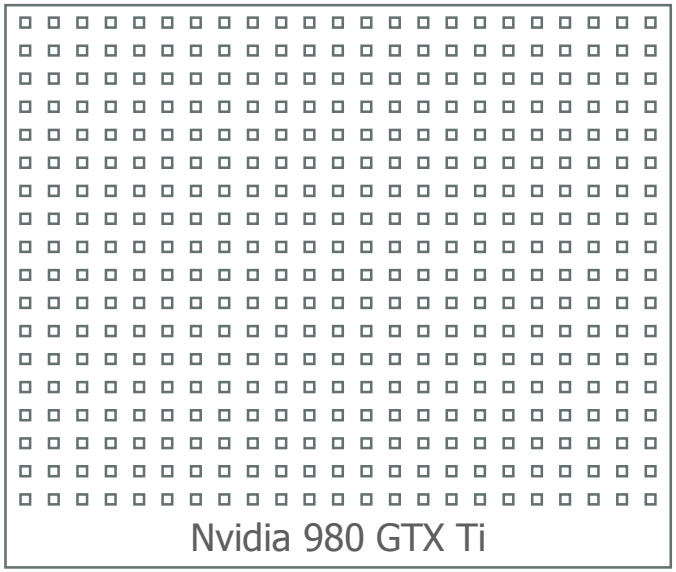
Quels outils pour le cassage de mots de passe ?

CPU vs GPU



CPU
- Traitement série
- Peu de cœurs
- Opérations complexes

GPU
- Traitement parallèle
- Nombreux cœurs
- Opérations simples



Outils classiques

/ Hashcat

/ John the ripper

/ Ophcrack

/ …

/ **02**

Nouvel outil de gestion des cassages

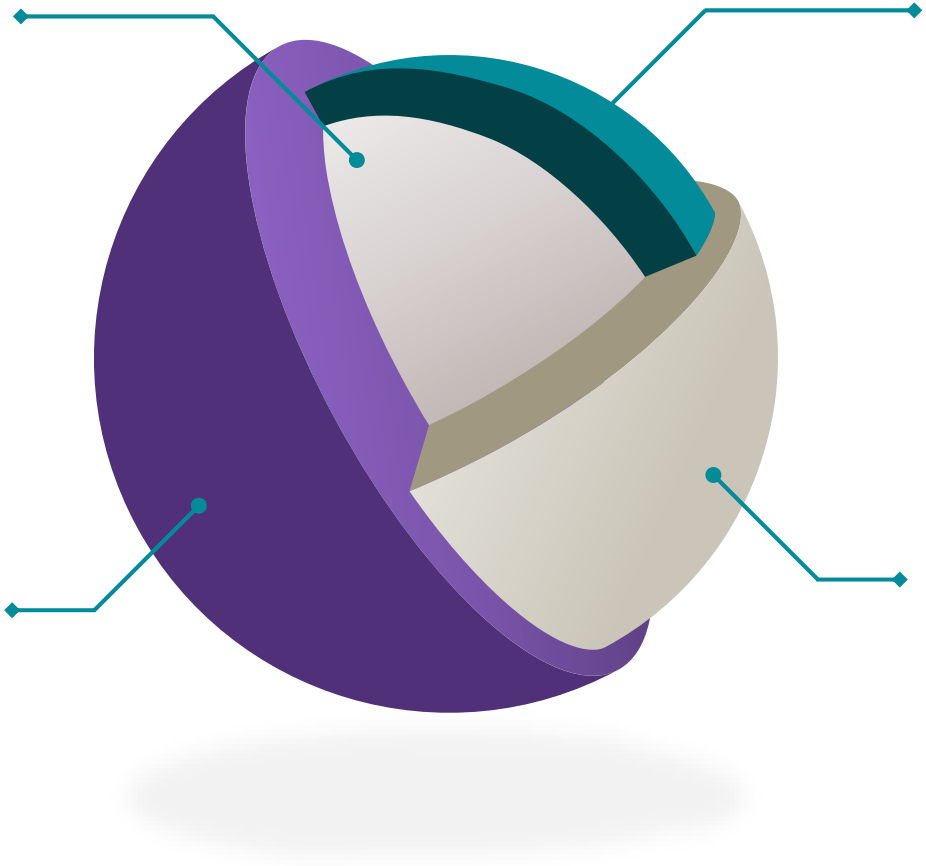
Wavecrack

Interface simplifiée

Interface web facilitant l'ajout de hash et sélection des modes de cassage

Statistiques

Production de statistiques sur les mots de passe cassés



Multi-utilisateurs

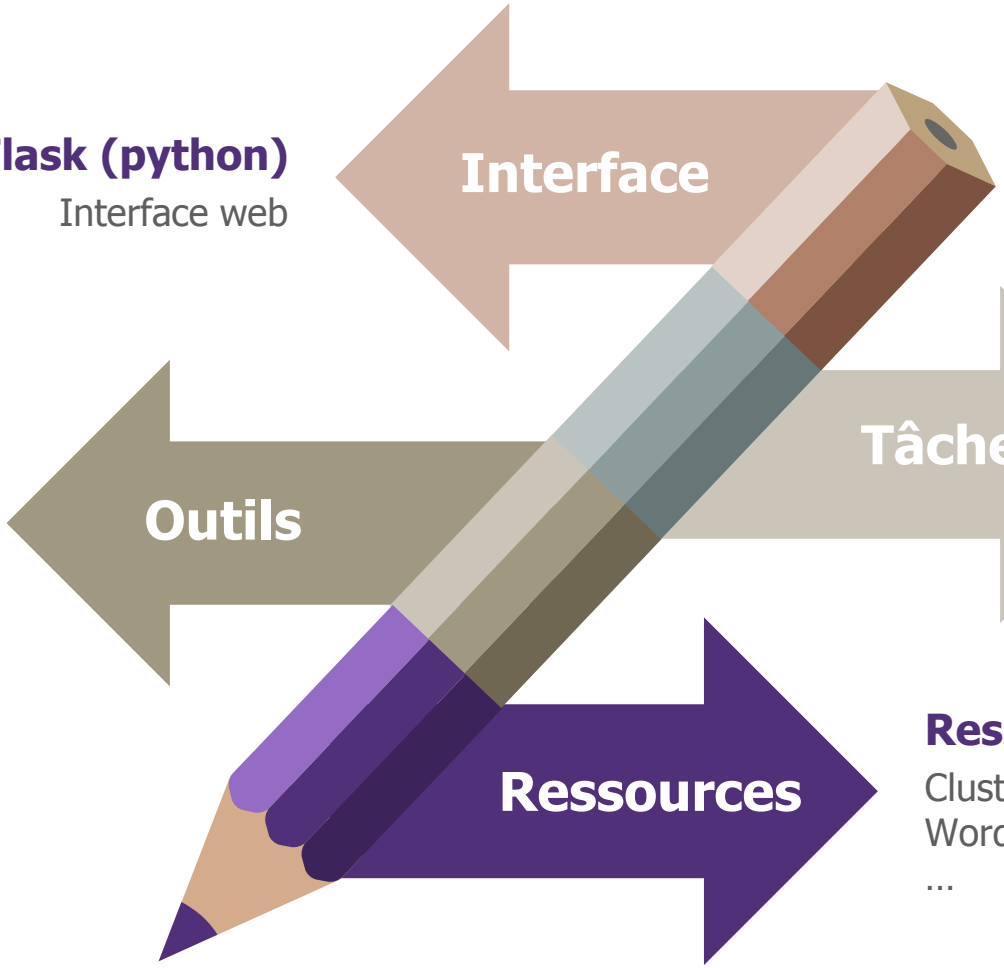
Cloisonnement des cassages entre les utilisateurs et partage des capacités de calcul

Développements personnalisés

Développements de fonctionnalités additionnelles à hashcat (gestion des hash LM, …)

Technologies

Flask (python)
Interface web



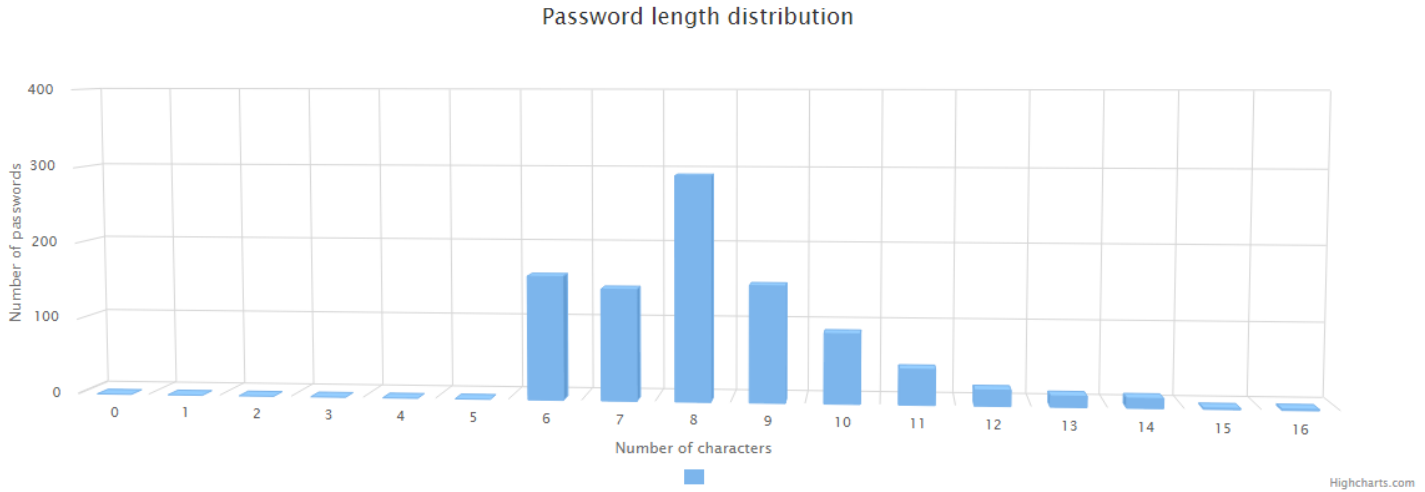
Celery
Permet de gérer les taches asynchrones

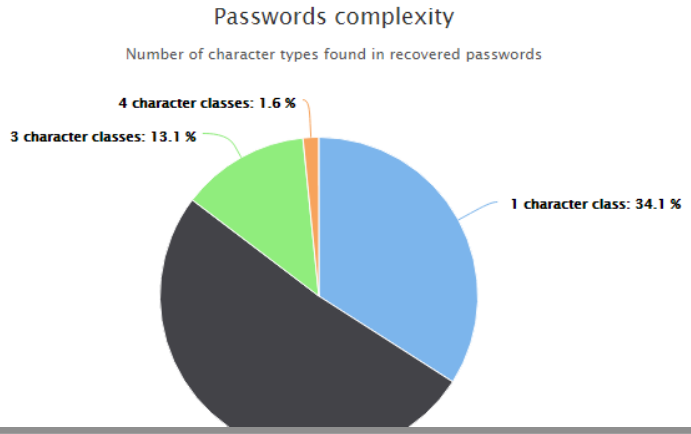
hashcat
Moteur de cassage de mots de passe

Ressources performantes
Cluster de GPU
Wordlists
...

Démonstration

000002a9ec1cc584ca70f5dd2146b1c12f821ac1	hagar12	5	0	2	0	Wordlist with variations (Previously cracked by Wavestone)





Retour d'expérience (sur 1 an)

TOP 20 des mots de passe cassés

1. 1 : 847 (1.49%)
2. P@ssw0rd!123 : 226 (0.40%)
3. PassPass2014 : 201 (0.35%)
4. Gravity2014 : 140 (0.25%)
5. [client] : 105 (0.19%)
6. : 80 (0.14%)
7. P@ssw0rd2015 : 60 (0.11%)
8. [projet]2012 : 59 (0.10%)
9. Password : 58 (0.10%)
10. password01. : 41 (0.07%)
11. [projet]2013 : 38 (0.07%)
12. Password2013 : 35 (0.06%)
13. 123456 : 32 (0.06%)
14. [client]123456! : 32 (0.06%)
15. [projet]2014 : 29 (0.05%)
16. 010203 : 28 (0.05%)
17. [client]1 : 28 (0.05%)
18. Bonjour01 : 24 (0.04%)
19. [projet]2015 : 24 (0.04%)
20. [projet]2013 : 23 (0.04%)

Méthodes les plus efficaces

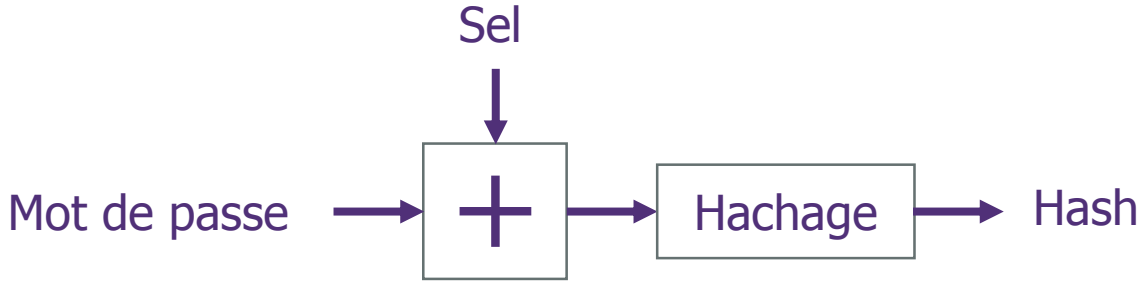
1. Dictionnaire : 37157 (65.50%)
2. Mots clés : 12465 (21.97%)
3. Bruteforce : 4918 (8.67%)
5. Masque : 2 (0.00%)

/ 03 Protections contre le cassage de hash

Protections

1. **Utiliser des algorithmes de hachage « lents » qui ralentissent les attaques**
   Exemples : bcrypt, PBKDF2

2. **Ajouter un sel unique pour chaque mot de passe**



➔ Les rainbow tables (précalculées) doivent être **générées pour chaque sel**

➔ Les attaques par bruteforce, dictionnaire, … **ne sont pas efficaces sur une grande liste de hash** de mots de passe à casser (extrait de base de données, de domaine Windows, etc.) :

- Chaque calcul de comparaison entre un mot de passe candidat et un hash ciblé devra être effectué autant de fois qu'il y a de mots de hash à casser.

Par exemple pour 10 000 hash, les performances du cassage seront divisées par 10 000

Protections

3

Politiques de mots de passe :

- **Une longueur importante est prioritaire sur l'utilisation de 4 classes de caractères**

$100^8 = 10^{16}$
$(26 + 26)^{12} = 4x10^{20}$ ➔ 10000 fois plus long à casser

- **Attention aux structures prédictibles** (mot suivi de quelques chiffres, leet, mots de passe courants, …)
  - ➢ Préférer l'utilisations de **phrases** où d'initiales de mots d'une phrase (paroles de chansons, phrase d'un livre, etc.) à des mots prédictibles (mots du dictionnaire, séquences de touches clavier, etc.)
  - ➢ Si possible, utiliser des mots de passe aléatoires et longs générés et stockés à l'aide d'un **générateur de mots de passe** (keepass, …)

4

Contrôler la robustesse des mots de passe côté serveur lors de leur définition

Notamment, vérifier que les mots de passe ne sont pas constitués de structures faibles (p.ex : « Mexique2016 », « Retraite2030 », « Client123 », etc.)

➔ Utilisation de « Password Filters » sur un Active Directory

WAVESTONE

Cyprien OGER
Consultant

T +33 (0)1 49 03 24 86 | **M**+33 (0)6 98 78 64 72
cyprien.oger@wavestone.com

wavestone.com
@wavestone_

PARIS

LONDRES

NEW YORK

HONG KONG

SINGAPORE *

DUBAI *

SAO PAULO *

LUXEMBOURG

MADRID *

MILAN *

BRUXELLES

GENEVE

CASABLANCA

ISTAMBUL *

LYON

MARSEILLE

NANTES

* Partenariats

WAVESTONE