Cryptanalyse - Chiffrement

Description

Le chiffrement est un procédé cryptographique qui consiste à rendre la compréhension d'un document ou d'une chaîne de caractères impossible à toute personne qui ne possède pas la clé de déchiffrement associée.

On distingue deux types de chiffrement :

- **Le chiffrement symétrique** qui repose sur une clé identique partagée entre les deux parties qui souhaitent communiquer, avec le même algorithme de chiffrement ;
- **Le chiffrement asymétrique** qui repose sur une paire de clés (privée et publique) : une clé pour chiffrer (clé publique) et une autre clé pour déchiffrer (clé privée), avec le même algorithme de chiffrement.

Les algorithmes de hashage sont différents des algorithmes de chiffrement dans la mesure où il est normalement impossible de revenir à l'origine (par exemple retrouver un mot de passe en clair depuis son hash), même en connaissant l'algorithme employé, la clé et tous les autres secrets de sa construction.

Les algorithmes de hashage sont des fonctions mathématiques à sens unique qui peuvent gérer les signatures numériques des documents afin d'assurer l'intégrité des données qu'ils contiennent, leur origine ainsi que leur authenticité.

Pré-requis d'exploitation

Pour exploiter cette méthode cryptographique, il est nécessaire d'avoir une connaissance approfondie de l'algorithme de chiffrement utilisé, ainsi que des compétences en programmation pour pouvoir l'implémenter dans un logiciel ou un script. Il est également important de comprendre les limites et les vulnérabilités potentielles de la méthode de chiffrement en question, ainsi que les attaques courantes qui peuvent être utilisées pour la compromettre. De plus, il peut être nécessaire d'avoir accès à la clé de chiffrement ou de disposer d'une méthode pour la déchiffrer.

Connaissances nécessaires

- Connaissance des différents algorithmes de chiffrement (AES, RSA, ...) et de leur niveau de sécurité ;
- Connaissance des techniques utilisées pour casser les algorithmes de chiffrement.

Outils nécessaires

- Outils en ligne (crackstation, md5decrypt) ou hors ligne (rsactftool, hashcat, johntheripper).

Flux d'exécution

Explorer

Lire la chaîne de caractères chiffrée ou le hash afin d'identifier l'algorithme qui l'a généré en analysant la taille de la chaîne et les caractères qu'elle comporte.

Expérimenter

Dans le cas d'un hash, tenter d'abord une recherche de ce hash à l'aide outils en ligne. En cas d'échec, réaliser une attaque par bruteforce avec des outils offline (dictionnaire avec des wordlists telles que **rockyou**. Enfin, en cas d'échec de cette méthode, tenter une attaque par bruteforce complet pour tenter de casser le hash.

Dans le cas d'une chaîne chiffrée, tenter différentes méthodes (factorisation, ECB, modules communs, ...) en fonction du type de chiffrement identifié (RSA, AES..).

Exploiter

Conséquences potentielles

Une exploitation réussie d'une vulnérabilité dans ce domaine peut permettre :

- La divulgation d'informations sensibles ;
- La compromission d'une base de données ou d'un serveur.

Contre-mesures

Les contre-mesures suivantes peuvent être mises en œuvre :

- Utiliser des algorithmes de chiffrement et de hachage conformes à l'état de l'art avec des sels aléatoires ;
- Sur les applications web, utiliser des fonctions connues de chiffrement, par exemple en PHP la fonction **password_hash** ;
- Éviter de développer des fonctions cryptographiques **maison** ;
- Utiliser un pare-feu applicatif pour bloquer les requêtes excessives ou suspectes afin d'empêcher les attaques par bruteforce.

Comment cela fonctionne ?

Le scénario suivant peut être joué via l'exploitation de cette vulnérabilité :

- Un attaquant réussi à accéder à une base de données contenant des informations sur les utilisateurs, telles que leurs noms d'utilisateur et leurs mots de passe stockés sous forme de hash. Pour pouvoir récupérer les mots de passe en clair, l'attaquant utilise alors une attaque par bruteforce pour essayer de trouver une correspondance entre les hashs et une liste de mots de passe courants. Après avoir essayé de nombreuses combinaisons de mots et de caractères, l'attaquant finit par trouver une correspondance et retrouve un mot de passe à l'origine d'un hash. Il peut alors se connecter avec les identifiants de l'utilisateur qu'il a récupéré et même usurper son identité.

Exemple 1

L'algorithme RSA va dans un premier temps générer deux couples de clés asymétriques, l'un pour l'émetteur qu'on appellera Alice, et l'autre pour le destinataire qu'on appellera Bob.

Une fois que chaque personne dispose de ses deux clés, ils peuvent établir une communication sécurisée. Alice va récupérer la clé publique de Bob (en général on pratique un échange des clés publiques avant de communiquer, ou alors on les diffuse publiquement) puis va chiffrer son message avec. Ensuite le message chiffré est transmis à Bob qui va le déchiffrer grâce à sa clé privée (qu'il n'a communiqué à personne).

Aucun échange de clé sensible n'est nécessaire, et seule la clé privée de Bob peut déchiffrer le message, la communication est alors sécurisée.

La sécurité de l'algorithme réside dans l'utilisation d'une fonction de chiffrement asymétrique, qui repose donc sur la confidentialité de la clé privée. Le déchiffrement sans la clé privée est complexe, mais pas nécessairement impossible si les algorithmes ou les tailles de clés sont trop faibles, ou mal implémentés.

Déchiffrement :

Si les normes de sécurité de l'algorithme RSA ne sont pas respectées, des attaques visant à retrouver ou à reconstituer la clé privée et ainsi déchiffrer le message chiffré sont possibles.

Les attaques par factorisation sur cet algorithme en sont un parfait exemple et peuvent être possibles à cause d'un choix de modulo trop petit, par exemple.

CWEs

- CWE-261 : Weak Encoding for Password
  - Obscuring a password with a trivial encoding does not protect the password.
- CWE-328 : Use of Weak Hash
  - The product uses an algorithm that produces a digest (output value) that does not meet security expectations for a hash function that allows an adversary to reasonably determine the original input (preimage attack), find another input that can produce the same hash (2nd preimage attack), or find multiple inputs that evaluate to the same hash (birthday attack).

References

URL :

- https://www.concours-alkindi.fr/docs/intro_crypto.pdf
- http://deptinfo.unice.fr/twiki/pub/Linfo/PlanningDesSoutenances20032004/blanc-degeorges.pdf
- https://perso.liris.cnrs.fr/nicolas.pronost/UCBL/CapesInfo/PrepaEcrit/AlgoProg/07-Chiffrement%20et%20cryptographie.pdf

Retour fiches vulnérabilités

- Cyber fiches vulnérabilités