Introduction à la cryptologie
Attaques génériques sur Su xMAC

2024-03

Classement

Ce TP est classé comme lecontrôle continude ce cours. Vous devez envoyer un rapport écrit (dans un format portable)détaillantvos réponses aux questions, et le code source correspondant, y compris tous les tests,avec instructions de compilation et d'exécutionavant le vendredi 5 avril, 18h00 (2024-04-05T18:00+0200) à :

pierre.karpman@univ-grenoble-alpes.fr.

Le travail en équipe de deux est autorisé mais pas obligatoire. Dans ce cas, un seul rapport doit être envoyé, avec les deux membres de l'équipe clairement identifiés.

F

L'utilisation de l'analyse dynamiquedésinfectants (à travers les options -fsanitize=adresseet - fsanitize=indéfini)est fortement encouragé pendant la phase de développement.

L'utilisation des optimisations du compilateur (via l'option -O3)est fortement encouragé lors de l'exécution d'attaques les plus coûteuses.

L'utilisation d'un logiciel d'intelligence artificielle à tout moment de ce travail est strictement interdite. En dehors de la bibliothèque standard C, vous êtes*pas*autorisé à s'appuyer sur des logiciels externes ou des fonctions de bibliothèque.

1 Le suffixeMACsmht48

La construction Su xMAC est une transformation générique d'une fonction de hachage en MAC. De manière informelle, étant donné une fonction de hachageH,le Su xMAC associéMest défini comme :

$$M(k, m) = H(m||k)$$

oùk (resp.m)est la clé (resp. le message) deMet·||·désigne la concaténation de chaînes.

Dans ce laboratoire, nous utiliserons un jouet Su xMACsmht48basé sur une fonction de hachage jouet \narrow-pipe Merkle-Damgard"ht48.La fonction de hachage est déjà implémentée et disponible surhttps://membres-ljk.imag.fr/Pierre.Karpman/ht48_2.tar.bz2, mais vous devez mettre en œuvre smht48toi-même.

Q.1 :Implémenter la fonctionsmht48de la signature et des spécifications suivantes :

```
/*
 * Entrée k : une clé de 48 bits stockée sous forme de tableau de 6 octets
 * Entrée blen : la longueur d'octet de m, stockée sur 64 bits
 * Entrée m : le message à hacher, dont la longueur doit être un
 ↪   nombre entier d'octets
 * Entrée h : espace réservé pour la balise résultante de 48 bits, à stocker sous forme de
 ↪   tableau de 6 octets. Doit avoir été attribué par l'appelant.
 * Sortie : void, h est écrasé avec le résultat ht48(m||k)
 * Attention : les octets clés de k doivent être ajoutés _dans l'ordre_ (k[0] en premier)
 */
videsmht48(const uint8_tk[statique6],uint64_tblen,const uint8_t
 ↪   m[blen],uint8_th[statique6]);
```

Q.2 :Vérifiez votre mise en œuvre desmht48sur levecteurs de testci-dessous. Vous pouvez utiliser le (déjà fourni)hachage d'impressionfonction pour imprimer la valeur du tag sur la sortie standard.

1. Valeur clé : {0,1,2,3,4,5} Valeur du message : {9,8,7,6,5,4} Valeur de la balise :EE75794547B8
2. Valeur clé : {0xE4,0x16,0x9F,0x12,0xD3,0xBA} Valeur du message : {9,8,7,6,5,4} Valeur de la balise : 5F265B72B5EC

2 Recherche exhaustive d'une clé légère

Nous souhaitons maintenant trouver une cléktel que pour la valeur du message {9,8,7,6,5,4}, on a une valeur de balise7D1DEFA0B8AD.Par hasard, nous sommes conscients du fait utile que (un tel possible) kn'a qu'un poids (en bits) de 7 (c'est-à-dire qu'il a exactement 7 bits définis sur un).[-]

Q.3 :

1. Implémenter une fonctionenregistrement de clésrechercherk.
2. Pour quelle(s) valeur(s) avez-vous trouvék?

Conseil:Une version raisonnablement bien implémentée de cette attaque prend environ 100 secondes pour s'exécuter sur un ordinateur portable moyen. Vous pouvez d'abord valider l'exactitude et l'e cacité de votre implémentation en recherchant une clé que vous connaissez, éventuellement de plus petit poids.

Q.4 :

1. Expliquez comment unrécupération de cléUne attaque comme celle-ci peut être utilisée comme étape préliminaire à une attaque de contrefaçon universelle.
2. L'inverse est-il toujours possible ? Autrement dit : une attaque de contrefaçon universelle conduit-elle toujours à une attaque de récupération de clé ?

[-] Ce type d'information pourrait éventuellement être obtenu à partir d'uncanal latéralattaque physique, mais en supposant que les touches soient échantillonnées uniformément, on aurait quand même de la chance d'en avoir une de poids seulement 7 !

3 contrefaçons existentielles issues de collisions

La conception de Su xMAC et le fait quesmht48est basé sur la fonction de hachage Merkle-Damgard à tube étroitht48permet d'utiliser des collisions pour les seconds pour obtenir des contrefaçons existentielles pour les premiers. Plus en détail, laissez la fonction de compression utilisée dansht48 être la fonctiontcz48_dmde signature :

```
/*
 * Entrée m : un bloc de message de 128 bits stocké sous forme de tableau de 16 octets
 * Entrée h : une valeur de chaînage de 48 bits stockée sous forme de tableau de 6 octets
 * Sortie : void, h est écrasé par le résultat
 */
videtcz48_dm(const uint8_tm[statique16],uint8_th[statique6]);
```

etIVdésigne la valeur initiale utilisée dansht48 (donné dansht48.h).Alors si les messages de 16 octets m1etm2sont telles que les valeurs calculées partcz48_dm(m1, IV)ettcz48_dm(m2, IV) sont les mêmes, on a ça pour toutes les clésk,les balises calculées parsmht48(k, 16, m1, h)et smht48(k, 16, m2, h)sont identiques.

Q.5 :

1. Expliquez pourquoi ce qui précède est vrai.

2. Comment cette propriété peut-elle être utilisée dans une attaque de contrefaçon existentielle poursmht48 ?

Q.6 :

1. Implémenter une fonctionrecherche collatéralequi calcule une collision de la forme ci-dessus pour le (déjà implémenté)tcz48_dmfonction de compression.

2. Implémenter une fonctionsmht48efqui dessine une clé de 48 bitskuniformément au hasard et qui utilise la collision danstcz48_dmpour obtenir une collision danssmht48du formulaire ci-dessus.

Conseil:Une version raisonnablement bien implémentée de la recherche de collision prend environ 4 secondes pour s'exécuter sur un ordinateur portable moyen. Une stratégie possible consiste à utiliser une structure de données "search" e cace, qui peut par exemple être implémentée avec unad hoctable de hachage de 2_{24}seaux.