Groupe de travail Réseau
Request for Comments : 2945
Catégorie : En cours de normalisation

T. Wu, Stanford University
septembre 2000
Traduction Claude Brière de L'Isle

Système SRP d'authentification et d'échange de clés

Résumé
Le présent document décrit un mécanisme cryptographiquement fort d'authentification de réseau connu sous le nom de protocole de mot de passe distant sécurisé (SRP, *Secure Remote Password*). Ce mécanisme convient pour négocier des connexions sûres en utilisant un mot de passe fourni par l'usager tout en éliminant les problèmes de sécurité traditionnellement associés aux mots de passe réutilisables. Ce système effectue aussi un échange sûr de clés dans le processus d'authentification, permettant que soient activées des couches de sécurité (protection de la confidentialité et/ou de l'intégrité) durant la session. Des serveurs de clés de confiance et des infrastructures de certificats ne sont pas requis, et les clients ne sont pas obligés de mémoriser ou gérer des clés à long terme. SRP offre à la fois les avantages de la sécurité et du déploiement sur les techniques de mise au défi/réponse, ce qui en fait un moyen de remplacement idéal lorsque on a besoin d'une authentification sûre par mot de passe.

1. Introduction

L'absence d'un mécanisme sûr d'authentification qui soit aussi facile à utiliser est un problème qu'on cherche à résoudre depuis longtemps avec la grande majorité des protocoles de l'Internet utilisés actuellement. Le problème possède deux aspects. Les utilisateurs aiment utiliser des mots de passe dont ils puissent se souvenir, mais la plupart des systèmes d'authentification fondés sur le mot de passe offrent peu de protection contre même les attaques passives, en particulier si on utilise des mots de passe faibles et faciles à deviner.

L'espionnage sur un réseau TCP/IP peut être réalisé facilement et très efficacement contre des protocoles qui transmettent les mots de passe en clair. Même les techniques dites de "mise au défi-réponse" comme celles décrites dans la [RFC2095] et la [RFC1760], qui sont conçues pour déjouer de simples attaques de "reniflage", peuvent être compromises par ce qu'on appelle une "attaque de dictionnaire". Cela survient lorsque un attaquant capture les messages échangés durant une session légitime du protocole et utilise ces informations pour vérifier une série de mots de passe devinés pris dans un "dictionnaire" pré compilé de mots de passe courants. Cela fonctionne parce que les usagers choisissent souvent des mots de passe simples, facile à mémoriser, qui sont aussi invariablement faciles à deviner.

De nombreux mécanismes existants exigent aussi que la base de données des mots de passe sur l'hôte soit gardée secrète parce que le mot de passe P ou un hachage confidentiel h(P) y est mémorisé et que la sécurité serait compromise si elle était divulguée. Cette approche dégénère souvent en "sécurité dans l'obscurité" et va à l'encontre de la convention UNIX de conserver un fichier "public" des mots de passe dont le contenu peut être révélé sans détruire la sécurité du système.

SRP satisfait aux exigences les plus strictes établies dans la [RFC1704] pour un protocole d'authentification sans divulgation. Il offre une protection complète contre les attaques aussi bien passives qu'actives, et l'accomplit efficacement en utilisant un seul cycle de calcul de style Diffie-Hellman, rendant possible son utilisation dans l'authentification interactive aussi bien que non interactive pour une large gamme de protocoles Internet. Comme il conserve sa sécurité lorsque il est utilisé avec des mots de passe à faible entropie, il peut être intégré directement dans les applications d'utilisateur existantes.

2. Conventions et terminologie

Le protocole décrit par le présent document est parfois désigné sous le nom de "SRP-3" pour des raisons historiques. Ce protocole particulier est décrit dans [SRP] et il est estimé qu'il a une très bonne résistance logique et cryptographique à la fois à l'espionnage et aux attaques actives.

Le présent document n'essaye pas de décrire SRP dans le contexte d'un protocole Internet particulier ; il décrit plutôt un protocole abstrait qui peut aisément s'adapter à une application particulière. Par exemple, le format spécifique des messages (y compris le bourrage) n'est pas spécifié. La décision sur ces questions a été laissée aux mises en œuvre du protocole.

La seule question de mise en œuvre qu'il vaut la peine de spécifier ici est la transposition entre les chaînes et les entiers. Les protocoles Internet sont en mode octet, alors que SRP effectue des opérations algébriques sur ses messages, de sorte qu'il est logique de définir au moins une méthode par laquelle les entiers puissent être convertis en une chaîne d'octets et vice versa.

Une chaîne S de n octets peut être convertie en un entier comme suit :

i = S[n-1] + 256 * S[n-2] + 256^2 * S[n-3] + ... + 256^(n-1) * S[0]

où i est l'entier et S[x] est la valeur du x^{eme} octet de S. En termes humains, la chaîne d'octets est l'entier exprimé en base 256, avec le chiffre de poids fort en premier. Lorsque on reconvertit en une chaîne, S[0] doit être différent de zéro (le bourrage est considéré comme étant un processus distinct, indépendant). Cette méthode de conversion convient pour le stockage de fichiers, la représentation en mémoire, et la transmission dans le réseau de valeurs de grands entiers. Sauf spécification contraire, on doit supposer cette transposition.

Si des mises en œuvre ont besoin de bourrer une chaîne qui représente une valeur d'entier, il est recommandé qu'elles utilisent des octets de zéros et les ajoutent au début de la chaîne. La reconversion en entier élimine automatiquement les octets à zéro en tête, rendant ce schéma de bourrage moins enclin à l'erreur.

La fonction de hachage SHA, lorsque elle est utilisée dans le présent document, se réfère à l'algorithme de résumé de message SHA-1 décrit dans [SHA1].

3. Mécanisme SRP-SHA1

Cette section décrit une mise en œuvre du protocole SRP d'authentification et d'échange de clés qui emploie la fonction de hachage SHA pour générer des clés de session et des preuves d'authentification.

L'hôte mémorise les mots de passe d'utilisateur comme des triplets de la forme :

{ <nom_d'utilisateur>, <vérificateur de mot de passe>, <sel> }

Les entrées de mot de passe sont générées comme suit :

<sel> = aléa()
x = SHA(<sel> | SHA(<nom_d'utilisateur> | ":" | <mot de passe brut>))
<vérificateur de mot de passe> = v = g^x % N

Le symbole | indique la concaténation des chaînes, l'opérateur ^ est l'opération d'exponentiation *(élévation à une puissance)*, et l'opérateur % est l'opération du reste entier. La plupart des mises en œuvre effectuent l'exponentiation et le reste dans une seule étape pour éviter de générer de lourds résultats intermédiaires. Noter que les 160 bits de résultat de SHA sont implicitement convertis en entier avant qu'il soit soumis à une nouvelle opération.

L'authentification est généralement à l'initiative du client.

Client		**Hôte**
U = <nom_d'utilisateur> -->		
	<--	s = <sel provenant du fichier de mot de passe>

S'étant identifié auprès de l'hôte, le client va recevoir le sel mémorisé chez l'hôte sous son nom d'utilisateur.

a = aléa()

```
A = g^a % N                 -->
                                        v = <vérificateur du mot de passe mémorisé>
                                        b = aléa()
                                <--     B = (v + g^b) % N
p = <mot de passe brut>
x = SHA(s | SHA(U | ":" | p))

S = (B - g^x) ^ (a + u * x) % N         S = (A * v^u) ^ b % N
K = SHA_Interleave(S)                   K = SHA_Interleave(S)
(cette fonction est décrite au paragraphe suivant)
```

Le client génère un nombre aléatoire, élève g à cette puissance modulo le champ du nombre premier, et envoie le résultat à l'hôte. L'hôte fait la même chose et ajoute aussi le vérificateur public avant de l'envoyer au client. Les deux côtés construisent alors la clé de session partagée sur la base de leurs formules respectives.

Le paramètre u est un entier non signé de 32 bits qui tire sa valeur des 32 premiers bits du hachage SHA1 de B, MSB en premier.

Le client DOIT interrompre l'authentification si B % N est zéro.

L'hôte DOIT interrompre la tentative d'authentification si A % N est zéro. L'hôte DOIT envoyer B après avoir reçu A du client, jamais avant.

À ce point, le client et le serveur devraient avoir une clé de session commune qui est sûre (c'est-à-dire, non connue des tiers). Pour finir l'authentification, ils doivent se prouver l'un l'autre que leurs clés sont identiques.

```
M = H(H(N) XOR H(g) | H(U) | s | A | B | K) ---->
                                        <----   H(A | M | K)
```

Le serveur va calculer M en utilisant son propre K et le comparer à la réponse du client. Si ils ne correspondent pas, le serveur DOIT interrompre et signaler une erreur avant qu'il tente de répondre à la mise au défi du client. Ne pas le faire pourrait compromettre la sécurité du mot de passe de l'utilisateur.

Si le serveur reçoit une réponse correcte, il produit sa propre preuve au client. Le client va calculer la réponse attendue en utilisant son propre K pour vérifier l'authenticité du serveur. Si le client a répondu correctement, le serveur DOIT répondre par sa valeur de hachage.

Les transactions dans cette description de protocole n'ont pas nécessairement une correspondance exacte avec les messages de protocole réels. Cette description n'est destinée qu'à illustrer les relations entre les différents paramètres et la façon dont ils sont calculés. Il est possible, par exemple, qu'une mise en œuvre du mécanisme SRP-SHA1 consolide certains des flux comme suit :

```
     Client                                     Hôte
U, A                    -->
                                        <--     s, B
H(H(N) XOR H(g) | H(U) | s | A | B | K) --->
                                        <--     H(A | M | K)
```

Les valeurs de N et g utilisées dans ce protocole doivent faire l'objet d'un accord de la part des deux parties en question. Elles peuvent être établies à l'avance, ou l'hôte peut les fournir au client. Dans ce dernier cas, l'hôte devrait envoyer les paramètres dans le premier message ainsi que le sel. Pour une sécurité maximum, N devrait être un premier sûr (c'est à dire un nombre de la forme N = 2q + 1, où q est aussi premier). Aussi, g devrait être un générateur modulo N (voir [SRP] pour les détails) ce qui signifie que pour tout X où 0 < X < N, il existe une valeur x pour laquelle g^x % N == X.

3.1 SHA entrelacé

La fonction SHA_Interleave utilisée dans SRP-SHA1 est utilisée pour générer une clé de session qui est de deux fois la longueur du résultat de 160 bits de SHA1. Pour calculer cette fonction, retirer tous les octets de zéro en tête du résultat. Si la longueur de la chaîne résultante est impair, retirer aussi le premier octet. Appelons la chaîne résultante T. Extraire les octets de numéro pair dans une chaîne E et les octets de nombre impair dans une chaîne F, c'est à dire ;

```
E = T[0] | T[2] | T[4] | ...
```

F = T[1] | T[3] | T[5] | ...

E et F devraient toutes deux être exactement la moitié de T. Hacher chacune avec le SHA1 régulier, c'est à dire :

G = SHA(E)
H = SHA(F)

Entrelacer les deux hachages ensemble pour former le résultat, c'est à dire :

résultat = G[0] | H[0] | G[1] | H[1] | ... | G[19] | H[19]

Le résultat aura 40 octets (320 bits) de long.

3.2 Autres algorithmes de hachage

SRP peut être utilisé avec des fonctions de hachage autres que SHA. Si la fonction de hachage produit un résultat d'une longueur différente de celle de SHA (20 octets) cela peut changer la longueur de certains des messages dans le protocole, mais le fonctionnement fondamental n'en sera pas affecté.

Des versions précédentes du mécanisme SRP utilisaient la fonction de hachage MD5, décrite dans la [RFC1321]. L'utilisation de transformations de hachage chiffré est aussi recommandée avec SRP ; une construction possible utilise HMAC [RFC2104], avec K pour chiffrer le hachage dans chaque direction au lieu de l'enchaîner avec les autres paramètres.

Toute fonction de hachage utilisée avec SRP devrait produire un résultat d'au moins 16 octets et avoir pour propriété que de petits changements dans l'entrée causent des changements significatifs non linéaires dans le résultat. [SRP] traite de ces questions plus en profondeur.

4. Considérations pour la sécurité

La totalité du présent mémoire discute d'un système d'authentification et d'échange de clés qui protège les mots de passe et les échanges de clés à travers un réseau qui n'est pas de confiance. Ce système améliore la sécurité en éliminant le besoin d'envoyer des mots de passe en clair sur le réseau et en activant le chiffrement à travers son mécanisme d'échange de clés sécurisé.

Les valeurs privées pour a et b correspondent en gros aux valeurs privées dans un échange Diffie-Hellman et elles ont des contraintes similaires de longueur et d'entropie. Les mises en œuvre peuvent choisir d'augmenter la longueur du paramètre u, pour autant que client et serveur en soient tous deux d'accord, mais il n'est pas recommandé qu'il fasse moins de 32 bits.

SRP n'a pas été conçu pour seulement contrer la menace de divulgation occasionnelle d'un mot de passe, mais aussi pour empêcher un attaquant déterminé équipé d'un dictionnaire de mots de passe de deviner les mots de passe en utilisant du trafic réseau capturé. Le protocole SRP lui-même résiste aussi aux attaques actives du réseau, et les mises en œuvre peuvent utiliser les clés échangées de façon sûre pour protéger la session contre la capture et assurer la confidentialité.

SRP présente aussi l'avantage supplémentaire de permettre à l'hôte de mémoriser les mots de passe sous une forme qui n'est pas directement utilisable pour un attaquant. Même si la base de données de mots de passe de l'hôte est révélée au public, l'attaquant aura encore besoin d'une coûteuse recherche de dictionnaire pour obtenir un mot de passe. Le calcul exponentiel nécessaire dans ce cas pour valider une conjecture est beaucoup plus consommateur de temps que le hachage actuellement utilisé par la plupart des systèmes UNIX. Il est cependant conseillé aux hôtes d'essayer de faire de leur mieux pour garder leurs fichiers de mots de passe à l'abri.

5. Références

[RFC1321] R. Rivest, "Algorithme de résumé de message MD5", avril 1992. *(Information)*

[RFC1704] N. Haller et R. Atkinson, "Authentification sur l'Internet", octobre 1994. *(Information)*

[RFC1760] N. Haller, "Système S/KEY de mot de passe à utilisation unique", février 1995. *(Information)*

[RFC2095] J. Klensin, R. Catoe, P. Krumviede, "Extension AUTHorize à IMAP/POP pour mise au défi/réponse simple", janvier 1997. (*Obsolète, voir* RFC2195) *(P.S.)*

[RFC2104] H. Krawczyk, M. Bellare et R. Canetti, "HMAC : Hachage de clés pour l'authentification de message", février 997.

[SHA1] National Institute of Standards and Technology (NIST), "Announcing the Secure Hash Standard", FIPS 180-1, U.S. Department of Commerce, avril 1995.

[SRP] T. Wu, "The Secure Remote Password Protocol", Dans Proceedings of the 1998 Internet Society Symposium on Network and Distributed Systems Security, San Diego, CA, pp. 97-111.

6. Adresse de l'auteur

Thomas Wu
Stanford University
Stanford, CA 94305
mél : tjw@cs.Stanford.EDU

7. Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'éditeur des RFC est actuellement assuré par la Internet Society.