Protocole SRP

Par Abdessalam BOULAHDID et Selim BEN AMMAR

Introduction

SRP (Secure Remote Protocol) est un mécanisme d'authentification réseau publié par l'université de Stanford en 2000. Plusieurs versions de ce protocole ont été développées dont SRP-6a (Le plus récent).

Toutefois on ne s'intéressera dans ce document qu'à la première version du protocole. On essayera d'abord d'analyser brièvement les protocoles d'authentification réseau existants à l'époque (Année 2000), puis on décrira le fonctionnement du SRP pour ensuite comprendre ce qu'il apporte en matière de sécurité.

I-Contexte du Développement de SRP

SRP a été pensé pour pallier aux problèmes de sécurité que présentaient les protocoles tout en étant facile à implémenter.
En effet, tout protocole d'authentification sert principalement à déterminer pour un serveur si une personne, qui doit avoir fourni une preuve d'authentification, peut ou pas accéder à un service. On peut classer ces protocoles en 3 grandes catégories selon le type de preuve:

- Une caractéristique de la personne (sa taille, sa démarche, ses empreintes…)
- Ce que possède la personne (une carte, une clé …)
- Ce que sait la personne (un identifiant, un mot de passe…)

Les protocoles d'authentification réseau font bien évidemment parties de la 3ème catégorie car c'est la moins coûteuse et la plus facile à déployer. Les couples identifiant/mot de passe sont largement utilisés pour ce type d'authentification, mais étant donné que ces informations seront envoyées dans un réseau pour ensuite être vérifiées par le serveur, il y a un risque qu'une tierce personne écoute le réseau pour compromettre l'échange de données client/serveur. Ainsi plusieurs protocoles ont vu le jours pour renforcer la sécurité des échanges sur le réseau et pour prévenir contre les attaques les plus courantes.

On supposera dans toute la suite que l'authentification se fait entre une personne (Alice) et un serveur (Bob) qui cherchent à échanger des informations sur le réseau. De plus on suppose l'existence d'une tierce personne (Eve) qui essaye de compromettre cet échange.

1 – Les techniques d'authentifications

i) La méthode la plus évidente est qu'Alice envoie un mot de passe à Bob, si celui-ci se trouve parmi les mots de passe que possède Bob, alors l'authentification est acceptée, sinon elle est rejetée. Ce protocole est très vulnérable aux attaques, on peut certes l'améliorer en envoyant le hash du mot de passe au lieu du mot de passe en clair, mais cela reste insuffisant car sensible au sniffing.

ii) Pour contrer cela, on peut utiliser ce qu'on appelle « Authentification défi-réponse », le scénario est le suivant :

1 - Alice envoie son identifiant à Bob + un message aléatoire.
2 - Bob envoie à Alice un message aléatoire appelé 'challenge'.
3 - Alice effectue des calculs basés sur le challenge en utilisant le message aléatoire et son mot de passe puis envoie le résultat à Bob qui fait de même et compare les deux.

Puisque Bob envoie un challenge différent à chaque authentification, Eve ne pourra pas utiliser les données sniffer pour d'autres sessions. Cependant ce type de protocole est tout de même vulnérable à d'autres types d'attaques, dont l'attaque par dictionnaire : Eve peut capturer le message aléatoire d'Alice et le challenge pour former des tentatives authentification acceptées par Bob, elle ne lui reste qu'à effectuer plusieurs tentatives en essayant plusieurs mot de passe jusqu'à trouver le bon. De plus ce protocole est « plaintext-

equivalent», cela veut dire que des données sensibles sont stockées par Bob dont la divulgation compromettra l'échange.

Étant donné les limitations intrinsèques de ce type de protocole, des chercheurs ont essayé plusieurs améliorations comme augmenter la taille du mot de passe ou encore rendre le sniffing plus difficile, néanmoins l'authentification reste faible, autrement dit facilement détourné par Eve.

iii) Parmi les protocoles d'authentification les plus sécurisés qui sont basés sur un mot de passe, on trouve les EKE qui utilisent, en plus d'un mot de passe, un échange de clé symétrique. De plus certains protocoles EKE utilisent un datage pour assurer une confidentialité persistante. Bien qu'à priori ce type de protocole soit robuste, il reste néanmoins vulnérable car c'est aussi un « plaintext-equivalent» .

2 – Une nouvelle technique d'authentification

Face à ces problématiques de sécurité, des chercheurs ont développé un nouveau mécanisme d'authentification se basant sur un 'Verifier', ce type de protocole est appelé « verifier-based protocol ». Contrairement au password-based protocol, Bob ne stocke plus les mots de passe mais plutôt des Verifiers calculés à l'aide des mots de passe en utilisant une fonction à sens unique. Cela résout notamment le problème des « plaintext-equivalent».

Ainsi, une famille de protocoles, basés sur les Verifiers et qui en plus répondent aux autres exigences de sécurité, a vu le jour: Les protocoles AKE (Asymetric key exchange) qui utilisent un échange asymétrique en plus d'un mot de passe et d'un Verifier. Parmi ces protocoles on trouve le SRP.

II-Le protocole SRP

1 – Spécification du protocole

Tous les calculs se font dans un corps GF(n) où n est un nombre premier, pour plus de détails on pourra se référer au RFC .
Les données qui seront utilisées par le protocoles sont les suivantes:

- n un grand nombre premier, tous les calcules sont modulo n.
- g une racine primitive modulo n (le générateur) .
- s (salt) une chaîne de caractère aléatoire.
- P le mot de passe d'Alice
- x une clé privée calculée à partir de P et s.
- v le Verifier associé à P .
- u un paramètre aléatoire publique
- a,b clés privés éphémères générés aléatoirement.
- A,B les clés publiques correspondantes.
- H() une fonction hachage a sens unique.
- K clé de la session

2 – Phase 1: Création du compte d'Alice

Alice choisit un mot de passe P, génère s, et calcule x = H(s,P) et v = g^x (on rappelle que tous les calcules sont modulo n).
Bob stocke v et s pour l'utilisateur Alice.

3 – Phase 2: Authentification

- Alice envoie son identifiant (Alice123 par exemple).
- Bob retrouve le couple (v,s) associé à Alice puis envoie ‘s’ à Alice.
- Alice calcule à nouveau x = H(s,P), puis génère le nombre ‘a’, calcule A = g^a et l’envoie à Bob.
- Bob génère le nombre ‘b’, calcule B = g^b + v et l’envoie à Alice en plus d’un paramètre u aléatoire.
- Alice et Bob calculent S = g^(ab + bux) en utilisant respectivement B et A, puis calculent le haché du résultat K=H(S).
- Alice envoie à Bob M1 = H(A,B,K) qui le vérifie.
- Si M1 est est validé, Bob envoie à Alice M2 = H(A,M(1), K) qui le vérifie.

4 – Analyse des étapes du protocole

i) On voit bien que le protocole est un « verifier-based protocol » puisque seul Alice est censée garder le mot de passe alors que Bob lui ne stocke que le couple (s,v). Ainsi si Eve récupère le couple (s,v), elle ne pourra pas usurper l’identité d’Alice à moins qu’elle réussisse à calculer l’inverse de g^x puis de H(s,P), ce qui est quasiment impossible puisque l’on connaît la difficulté du calcul du logarithme discret. Toutefois ce calcul est faisable dans certains cas : si n-1 est un produit de petit nombre premier, l’algorithme de Silver-Pohlig-Hellman est assez efficace. Pour contrer cela il faut choisir un nombre premier sûr, c’est à dire n = 2p+1 ou p est lui même premier.
Bien entendu, le couple (s,v) permet de se passer pour Bob, il faut donc tout de même le garder secret. De plus il est évident que si Eve récupère l’identifiant et le mot de passe d’Alice, elle pourra se faire passer pour elle, cela est un problème général de la 3eme catégorie des protocoles d’authentification qu’on ne peut résoudre.

ii) Pourquoi choisir B = g^b +v au lieu de B = g^b pour simplifier le protocole? La raison est simple, si on n’inclut pas v dans l’expression de B (B = f(g^b,v)), Eve peut utiliser une attaque par dictionnaire.
En effet, supposons que B = g^b. Eve va procéder comme suit:

- Eve récupère d’abord le salt en écoutant le réseau (ce qui est facile à faire).
- Elle se fait passer pour Bob (par usurpation d’IP ou autre) et commence l’échange avec Alice.
- Alice envoie à Eve son identifiant, puis Eve lui envoie à son tour le salt qu’elle a récupéré auparavant.
- Alice envoie à Eve A = g^a.
- Eve choisit son b et u, calcule son B = g^b et envoie B et u à Alice.
- Alice calcule la clé de session K (K = H(S) où S = (B-g^x)^(a+ux)) et l'envoie à Eve.
- Eve interrompt la connexion pour une raison quelconque (erreur…).

Maintenant que Eve possède K, A, b et u, elle peut choisir un mot de passe P', calculer x',
S' = (B-g^x')^(a+ux') puis K' =H(S') et comparer avec K jusqu'à trouver le bon. Ainsi elle peut trouver le mot de passe P par attaque de dictionnaire. Cette attaque suppose bien sûr que Eve ne connaît pas v car sinon elle aura pu faire directement l'attaque sur v = g^x.
On voit donc bien l'importance du v dans l'expression de B = f(g^b,v), de plus on veut que B ne révèle aucune information sur v, d'où le choix de f(x,y) = x+y pour la première version du SRP.

Le choix de la fonction f(x,y) = x+y présente toutefois un défaut. En effet cela permet à Eve de deviner le mot de passe d'Alice deux fois plus rapidement comme suit :

- Eve se fait passer pour Bob et commence l'échange avec Alice (elle ne connaît pas le couple (s,v))
- Arrivé à l'étape du calcule de B, Eve devine deux mots de passe P1 et P2 et calcule x1 et x2 puis envoie à Alice B = g^x1 + g^x2 (autrement dit elle n’utilise pas une valeur b aléatoire)
- Alice calcule la clé K en utilisant la valeur B – v = B – g^x, donc si x1 = x ou que x2 = x on aura

respectivement B = g^x2 ou B = g^x1.

- Eve calcule deux clés K1 et K2 en utilisant x1 et x2 et compare avec K, si l'un deux correspond à la clé, le xi utilisé (i =1 ou 2) donne le mot de passe d'Alice Pi (i=1 ou 2).

Ainsi Eve effectue deux tests en une tentative de connexion.
Dans SRP-6, on a pris f(g^b,v) = 3*v + g^b pour pallier à ce problème (le choix du nombre 3 dans l'expression est justifié mathématiquement). Cependant il s'est avéré que cela ne suffit pas, ce qui a amené à remplacer le 3 dans SRP -6a par un choix plus sûr: H(g,n).

iii) Que peut bien être le rôle de u dans le protocole puisque cette valeur est envoyée en clair à Alice?
Supposons que Eve a pu récupérer (par exemple en s'introduisant dans le serveur de Bob) le verifier v d'Alice, et supposons de plus que elle a la capacité de prédire u avant que celui ci ne soit envoyé (par exemple si le u est fixé ou que le générateur de nombre aléatoire employé est facilement prédictible), alors Eve peut usurper l'identité de Alice comme suit :

- Eve envoie l'identifiant d'Alice à Bob et reçoit le salt s.
- Eve calcule A = g^a.v^(-u) (normalement A = g^a) et l'envoie à Bob
- Bob envoie son B = v + g^b.
- Eve calcule la clé K comme : K = H((B-v)^a).
- Bob lui calcule la clé K' = H((A.v^u)^b) or (A.v^u)^b = g^ab et donc K = K'.

Ainsi Bob accepte l'authentification d'Eve comme étant Alice, d'où l'importance de la valeur u. Cette valeur ne doit être révélée qu'après l'envoie du A par Alice (comme fait dans le protocole) et ne doit pas être prédictible. On peut résoudre ce problème en générant par exemple le u à partir du A et du B (le u ne doit bien entendu pas prendre des valeurs triviales comme u = 0).

Conclusion:

Le protocole SRP est un protocole d'authentification qui résiste aux attaques les plus courantes (attaque par dictionnaire, attaque sur un «plaintext-equivalent», sniffing….) comme démontré plus haut. De plus il est facile à implémenter et ne requiert pas d'autres entités pour l'authentification (Exemple Kerberos où l'on a besoin entre autres d'un serveur d'émission de tickets et d'un centre de distribution de clés).
La dernière version du protocole (SRP 6a) est utilisée notamment pour l'authentification SSL/TLS (TLS-SRP) et a été standardisé dans le IEEE P1363 et ISO/IEC 11770-4.

Webographie:

- http://srp.stanford.edu/
- http://srp.stanford.edu/ndss.html
- https://www.ssi.gouv.fr/archive/fr/politique_produit/catalogue/pdf/authentification_robustesse_standard_v0-13.pdf
- https://en.wikipedia.org/wiki/Secure_Remote_Password_protocol
- http://srp.stanford.edu/srp6.ps