Groupe de travail Réseau
Request for Comments : 4387
Catégorie : Sur la voie de la normalisation

P. Gutmann, éditeur, University of Auckland
février 2006
Traduction Claude Brlière de L'Isle

Protocoles de fonctionnement d'infrastructure de clé publique X.509 pour l'Internet : Accès aux mémoires de certificats via HTTP

Statut de ce mémoire
Le présent document spécifie un protocole en cours de normalisation de l'Internet pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Normes officielles des protocoles de l'Internet" (STD 1) pour connaître l'état de la normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Les conventions de protocole décrites dans le présent document satisfont à certaines des exigences de fonctionnement de l'infrastructure de clé publique Internet (PKI, *Internet Public Key Infrastructure*). Le document spécifie les conventions pour utiliser le protocole de transfert hyper texte (HTTP/HTTPS, *Hypertext Transfer Protocol*) comme mécanisme d'interface pour obtenir des certificats et des listes de révocation de certificat (CRL, *certificate revocation list*) des répertoires PKI. Des mécanismes supplémentaires visant les exigences du fonctionnement de PKIX sont spécifiées dans d'autres documents.

Table des matières

1. Introduction

La présente spécification fait partie d'une norme multi parties sur l'infrastructure de clé publique Internet (PKI, *Internet Public Key Infrastructure*) utilisant les certificats X.509 et les listes de révocation de certificat (CRL). Le document spécifie les conventions pour l'utilisation du protocole de transfert hyper texte (HTTP, *Hypertext Transfer Protocol*) ou facultativement HTTPS, comme mécanisme d'interface pour obtenir des certificats ou des clés publiques, et des listes de

révocation de certificat (CRL) des répertoires PKI. Dans la suite du document le terme générique HTTP sera utilisé pour couvrir l'une et l'autre option.

Bien que la [RFC2585] couvre la récupération de certificats via HTTP, elle mentionne simplement que les certificats peuvent être récupérés sur un URL statique, qui ne fournit aucune capacité d'interface d'utilisation générale pour un magasin de certificats. Les conventions décrites dans le présent document permettent d'utiliser HTTP comme interface transparente d'utilisation générale, pour tout type de magasin de certificats ou de clés incluant des fichiers plats, des bases de données standard comme Berkeley DB et des bases de données relationnelles, et les répertoires traditionnels X.500/LDAP. Les applications typiques vont inclure l'utilisation avec des bases de données relationnelles ayant la capacité d'aller sur la Toile (ce que sont la plupart des bases de données) ou de simples mécanismes de recherche {clé, valeur} comme Berkeley DB et ses divers descendants.

Les mécanismes supplémentaires qui visent les exigences du fonctionnement de PKIX sont spécifiés dans des documents distincts.

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

2. Interface de mémorisation de certificats HTTP

La méthode GET est utilisée combinée avec unURI d'interrogation HTTP [RFC2616] pour restituer les certificats à partir du magasin de certificats sous-jacent :

```
http_URL = "http:" "//" hôte [ ":" accès ] [ chemin_abs [ "?" interrogation ]]
```

Les paramètres pour la portion interrogation de l'URI sont un certificat ou un identifiant de clé consistant en un type d'attribut et une valeur qui spécifie un ou plusieurs certificats ou clés publiques à retourner dr l'interrogation :

```
interrogation = attribut '=' valeur
```

Les certificats et clés publiques sont restitués d'un URI (l'URL de certificat) et les CRL d'un autre URI (l'URI de révocation). Ils peuvent ou non correspondre au même magasin de certificats et/ou serveur (l'interprétation exacte est une affaire de configuration locale). La valeur d'interrogation DOIT être codée en utilisant le type de support codé en format d'url [RFC2854]. Plus de détails sur la construction d'URI, ses limites de taille, et autres facteurs se trouvent dans la [RFC2616].

Les réponses aux interrogations infructueuses (par exemple, pour indiquer une non correspondance ou une condition d'erreur) sont traitées de la façon habituelle conformément à la [RFC2616]. Les clients devraient en particulier savoir que dans certaines instances, les serveurs peuvent retourner des demandes de redirection de type HTTP 3xx pour rediriger explicitement des interrogations sur un autre serveur. Évidemment, une redirection implicite fondée sur le DNS est aussi possible.

Si plus d'un certificat correspond à une interrogation, ils DOIVENT être retournés comme réponse multiparties/mixte. Les données retournées DOIVENT être retournées telles qu'elles ; elles NE DOIVENT PAS utiliser de contenu supplémentaire ou de codage de transfert au niveau HTTP (par exemple, elle ne peuvent pas être compressées ou codées en base64 ou texte quoted-printable). Les mises en œuvre NE DEVRAIENT PAS utiliser de codage de tronçons dans les réponses.

Le composant d'interrogation de l'URI PEUT facultativement contenir des paires d'attribut/valeur supplémentaires séparées par le délimiteur standard éperluette "&" qui spécifie que plus d'actions vont être entreprises par le magasin de certificats. Les magasins de certificats DEVRAIENT ignorer toute paire supplémentaire d'attribut/valeur non reconnue présente dans l'URI.

D'autres informations, comme les conventions de désignation et les types MIME, sont spécifiées dans la [RFC2585] (avec des types MIME supplémentaires pour les contenus non X.509 dans les [RFC3156] et [RFC3275]).

2.1 Conversion de gouttes binaires en clés de recherche

Certains champs (indiqués par la colonne "Traitement" dans les tableaux ci-dessous) sont de longueur arbitraire et/ou

contiennent des données non textuelles. Ces deux propriétés les rendent inaptes à l'utilisation directe dans les interrogations HTTP. Pour les rendre utilisables, les champs pour lesquels l'option de traitement est "Hash" sont d'abord hachés en une valeur de longueur fixe de 160 bits. Les champs pour lesquels l'option de traitement est "Hash" ou "Base64" sont codés en base64 pour transformer les données binaires en formes textuelles :

Option de traitement	Étape de traitement
"Hash"	Hache la valeur de clé en utilisant SHA-1 [FIPS180] pour produire une valeur de 160 bits, puis continue avec l'étape de codage base64 qui suit.
"Hash" "Base64"	Code la valeur binaire en utilisant le codage base64 pour produire une valeur seulement de texte de 27 octets. Le codage Base64 de la valeur de 20 octets va produire 28 octets, et le dernier octets va toujours être un caractère de bourrage "=". La valeur de 27 octets est créée en éliminant le caractère "=" en queue.

Pour les cas où la valeur binaire est inférieure ou supérieure au résultat de 20 octets de SHA-1 (par exemple, avec des identifiants de clé PGP de 64 bits/8 octets) la valeur finale est créée en supprimant tout le bourrage de "=" en queue du codage de la valeur binaire (c'est une généralisation du cas ci-dessus).

Les mises en œuvre DOIVENT vérifier que les valeurs codées en base64 soumises dans les demandes contiennent seulement des caractères dans les gammes 'a'-'z', 'A'-'Z', '0'-'9', '+', et '/'. Les interrogations qui contiennent tout autre caractère DOIVENT être rejetées. (Voir les notes de mise en œuvre du paragraphe 2.5 et les considérations sur la sécurité à la Section 4 pour plus de détails sur cette exigence.)

2.2 Types d'attribut : X.509

Les types d'attribut permis et leurs valeurs associées pour être utilisés avec des certificats X.509 et des listes de révocation sont décrits ci-dessous. Des valeurs binaires de longueur arbitraire (comme indiqué dans le tableau ci-dessous) sont converties en une clé de recherche par le processus décrit au paragraphe 2.1. Noter que les valeurs sont vérifiées pour correspondance exacte (après le décodage de toute portion codée en format d'url [RFC2854] si nécessaire) et sont donc sensibles à la casse.

Attribut	Traitement	Valeur
certHash	Hash	Clé de recherche déduite du hachage SHA-1 du certificat (parfois appelé empreinte digitale de certificat).
uri	aucun	URI sujet associé au certificat, sans le spécificateur (facultatif) de schéma. Le type d'URI dépend du certificat. Pour les certificats S/MIME, ce serait une adresse de messagerie électronique ; pour SSL/TLS, ce serait le nom DNS du serveur (c'est généralement aussi spécifié comme CommonName) ; pour IPsec, ce serait le nom DNS/adresse IP, et ainsi de suite.
iHash	Hash	Clé de recherche déduite du codage en DER du nom de domaine du producteur comme il apparaît dans le certificat, la CRL, ou autre objet.
iAndSHash	Hash	Clé de recherche déduite du codage en DER du issuerAndSerialNumber *(nom du producteur et numéro de série)* [RFC3852] du certificat.
name	aucun	Nom commun sujet contenu dans le certificat.
sHash	Hash	Clé de recherche déduite du nom de domaine sujet codé en DER comme il apparaît dans le certificat ou autre objet.
sKIDHash	Hash	Clé de recherche déduite de l'identifiant de clé sujette du certificat (en particulier les octets du contenu de la chaîne d'octets Identifiant de clé).

Les URI de certificat DOIVENT prendre en charge la restitution par tous les types d'attribut ci-dessus.

Les URI de CRL DOIVENT prendre en charge la restitution par les types d'attribut iHash et sKIDHash, qui identifient le producteur de la CRL. De plus, les URI de CRL PEUVENT prendre en charge la restitution par les types d'attribut certHash et iAndSHash, pour les cas où c'est exigé par l'utilisation de l'extension issuingDistributionPoint. Une interrogation de CRL DOIT retourner la CRL correspondante avec la plus grande valeur de thisUpdate (en d'autres termes, la plus récente CRL).

2.3 Types d'attribut : PGP

Les types d'attribut permis et leurs valeurs associées pour l'utilisation avec les clés publiques PGP et les informations de

révocation sont décrits ci-dessous. Les valeurs binaires (indiquées dans le tableau ci-dessous) sont converties en une clé de recherche par le processus décrit au paragraphe 2.1.

Attribut	Traitement	Valeur
email	aucun	adresse de messagerie électronique associée à la clé.
fingerprint	Base64	empreinte digitale de clé PGP de160 bits [RFC2440].
keyID	Base64	identifiant de clé PGP de 64 bits [RFC2440].
name	aucun	nom d'utilisateur associé à la clé.

Les URI de clés DOIVENT prendre en charge la restitution par tous les types d'attribut ci-dessus.

Les URI de révocation DOIVENT prendre en charge la restitution par les types d'attribut fingerprint et keyID, qui identifient le producteur de la révocation de clé.

2.4 Types d'attribut : XML

Les types d'attribut permis et leurs valeurs associées pour l'utilisation avec XML sont spécifiés aux paragraphes 2.2 et 2.3. Comme XML permet d'associer des attributs arbitraires à l'élément fils <RetrievalMethod> (*méthode de restitution)* de <KeyInfo> [RFC3275], il n'y a pas d'exigence supplémentaire particulière à utiliser avec XML.

2.5 Notes de mise en œuvre et raisons

Ces paragraphes pour information précisent les raisons qui sous-tendent les concepts de la Section 2 et donnent des lignes directrice pour la mise en œuvre.

2.5.1 Identification

Les identifiants sont tirés de PKCS n° 15 [PKCS15], une norme qui couvre (entre autres choses) une interface transparente avec le magasin de certificats/clés publiques. Ces identifiant ont subi l'épreuve du feu, car il ont été d'usage courant pendant un certain nombre d'années, normalement via PKCS n° 11 [PKCS11]. Les magasins de certificats et les identifiants requis pour les opérations normales de recherche de certificat sont analysées plus en détails dans [Gutmann].

Le type d'identifiant d'URI spécifie l'identifiant associé à l'usage prévu du certificat avec un certain protocole de sécurité de l'Internet. Par exemple, un certificat de serveur SSL/TLS va contenir le nom DNS du serveur (ce qui est traditionnellement aussi spécifié comme nom commun ou CN) un certificat S/MIME va contenir l'adresse de messagerie électronique du sujet, un certificat IPsec va contenir un nom DNS ou une adresse IP, et un certificat SIP va contenir un URI SIP. Un minimum de bon sens est supposé pour décider de la valeur de champ d'URI appropriée.

Pour des raisons historiques qui remontent à sa première utilisation comme moyen de chercher les certificats de messagerie électronique S/MIME des utilisateurs, certains clients peuvent spécifier le nom d'attribut d'URI comme "email" plutôt que "uri". Bien que ce ne soit pas exigé par la présente spécification, les serveurs peuvent choisir de permettre l'utilisation de "email" comme alias pour "uri".

De plus, il est de pratique courante d'utiliser l'identifiant Internet associé au champ d'application prévu du certificat comme CN pour le certificat quand c'est le nom le plus parlant pour le sujet du certificat. Par exemple, un certificat de serveur SSL/TLS va contenir le nom DNS du serveur dans le champ CN. Dans les appareils qui ont une capacité d'accès à la Toile, ceci peut bien sûr être le seul nom qui existe pour l'appareil. Il est donc assez possible que l'URI duplique le CN, et qu'il puisse être le seul identifiant présent (c'est-à-dire qu'il n'y a pas un DN complet mais seulement un seul champ CN).

Par une convention établie depuis longtemps, les URI dans les certificats sont donnés sans spécification de schéma. Par exemple, un certificat de serveur SSL/TLS va contenir www.exemple.com plutôt que https://www.exemple.com, et un certificat S/MIME va contenir user@exemple.com plutôt que mailto:user@exemple.com. Cette convention est aussi étendue aux autres types d'URI, de sorte qu'un certificat contenant les URI (effectifs) im:user@exemple.com et xmpp:user@exemple.com sera interrogé en utilisant le seul URI user@exemple.com. Le magasin de certificats va alors retourner tous les certificats contenant cet URI, laissant au client le soin de déterminer lequel est le plus approprié pour son usage. Cette approche est retenue à la fois parce que pour la plupart des types d'URI courants, il n'y a pas de spécification de schéma (voir les paragraphes précédents) et pas de moyen facile pour déterminer quel est l'usage prévu (un certificat de serveur SSL/TLS est simplement celui présenté par un serveur SSL/TLS) et parce que la partie/client demandeur est dans une meilleure position que le serveur du magasin de certificats pour juger de l'utilisation la plus appropriée du certificat.

Un autre identifiant possible qui a été suggéré est une adresse IP ou un nom DNS, qui va être exigé pour les appareils à capacité d'accès à la Toile incorporée. Ceci est nécessaire pour permettre par exemple d'interroger un contrôleur automatique de rattachement pour des certificats pour les appareils qu'il contrôle. Comme cette valeur est considérée comme le CN pour l'appareil, la pratique courante est d'utiliser cette valeur comme CN de la même façon que les certificats de serveur de la Toile règlent le CN au nom DNS du serveur, de sorte que cette option est déjà couverte d'une façon tout à fait acceptée.

Le nom et l'adresse de messagerie sont une exacte copie de ce qui est présent dans le certificat, sans aucune canonisation ni réécriture (autre que le codage de transport requis par HTTP). Cela suit la pratique des mises en œuvre standard, qui transfère une exacte copie de ces éléments de données afin d'éviter les problèmes dus à la traduction de jeu de caractères, au traitement des espaces, et autres.

Les hachages sont utilisés pour des champs de longueur arbitraire comme ceux qui contiennent des DN à la place du champ complet pour que la longueur reste gérable. De plus, l'utilisation de la forme hachée souligne que la recherche de données de nom structurées n'est pas une caractéristique prise en charge, car c'est une simple interface à un magasin de certificats {clé,valeur} et non une interface HTTP à un répertoire X.500. Les utilisateurs qui exigent spécifiquement une interface HTTP à X.500 peuvent utiliser une technologie comme celle des passerelles HTTP LDAP à cette fin.

Bien que les clients soumettent toujours une valeur fixe de 160 bits, les serveurs sont libres d'utiliser autant de bits de cette valeur que nécessaire. Par exemple, un serveur peut choisir d'utiliser seulement les 40, 64, 80, ou 128 premiers bits pour l'efficacité de la recherche et la conservation des indices.

PGP a traditionnellement codé les identifiants en utilisant une notation 0xABCDEF de style C fondée sur le format d'affichage utilisé pour les identifiants dans PGP 2.0. Malheureusement, les chaînes de ce format sont aussi des chaînes valides dans le format base64, compliqué du fait que des presque correspondances comme 0xABCDRF pourraient être soit une tentative avec une faute de frappe d'un identifiant hexadécimal soit un identifiant valide base64. Pour cette raison, et pour assurer la cohérence, les identifiants base64 sont utilisés tout au long de la présente spécification. Les clés de recherche utilisées en interne vont être des valeurs binaires, de sorte qu'elles soient converties de l'hexadécimal ASCII ou du base64 est sans importance à long terme.

Les attributs reçoivent des formes de nom abrégées (par exemple, iAndSHash à la place de issuerAndSerialNumberHash) afin de garder des longueurs raisonnables, ou des formes de nom commun (par exemple, email à la place de rfc822Name, rfc822Mailbox, emailAddress, mail, ou email) lorsque plusieurs formes de nom existent.

Dans certains cas, les utilisateurs peuvent exiger des types supplémentaires d'attributs, spécifiques de l'application. Par exemple, une application médicale qui utilise un identifiant médical comme clé principale pour ses bases de données peut avoir besoin de la capacité d'effectuer des recherches de certificat sur la base de cet identifiant médical. Le formatage et l'utilisation de tels identifiants spécifiques d'application sort du domaine d'application de ce document. Cependant, ils devraient commencer par 'x-' pour assurer qu'ils n'entrent pas en conflit avec les identifiants qui pourraient être définis dans de futures versions de cette spécification.

2.5.2 Vérification des valeurs d'entrée

Il devrait être attentivement vérifié que la portion valeur d'attribut de l'identifiant ne contient pas de caractères invalides car permettre des données brutes présente un risque pour la sécurité. Considérons, par exemple, un magasin de certificats/clés publiques mis en œuvre en utilisant un RDBMS dans lequel l'interrogation SQL est construite comme "SELECT certificate FROM certificates WHERE iHash = " + <clé de recherche>. Si <clé de recherche> est réglé à "ABCD;DELETE FROM certificates", le résultat de l'interrogation va être assez différent de ce qui était attendu par l'administrateur du magasin de certificats. Même une interrogation en lecture seule peut être problématique ; par exemple, régler <clé de recherche> à "UNION SELECT password FROM master.sysxlogins" va faire la liste de tous les mots de passe dans une base de données de serveur SQL (dans un format facile à déchiffrer) si l'utilisateur fonctionne avec le compte sa (DBA). Pour cette raison, seuls les codages valides en base64 devraient être permis. La même vérification s'applique aux interrogations par nom ou adresse de messagerie électronique.

Une vérification directe de la bonne santé des interrogations peut n'être pas suffisante pour empêcher toutes les attaques ; par exemple, un filtre qui supprime la chaîne d'interrogation SQL "DELETE" peut être outrepassé en soumettant la chaîne incorporée dans une autre instance de la chaîne. Supprimer "DELETE" de "DELDELETEETE" laisse en place le "DELETE" externe. Tromper avec des filtres la troncature de chaînes très longues peut aussi être utilisé comme moyen d'attaque, l'attaquant s'assurant que la troncature se produit au milieu d'une séquence d'échappement, outrepassant le

filtrage. Bien qu'en théorie le filtrage récurrent puisse aider ici, l'utilisation d'interrogations paramétrées (souvent appelées bouche trous) qui ne sont pas vulnérables aux injections de SQL devrait être utilisée pour éviter ces attaques. Plus d'informations sur la sécurisation des extrémités de bases de données se trouvent dans [Birkholz], et des commentaires sur la vérification de bonne santé et les problèmes de sécurité se trouvent dans la section de considérations sur la sécurité.

2.5.3 Notes sur les URI

Les URI pré-construits qui vont chercher une correspondance de certificat/clé publique avec un critère de recherche fixé peuvent être utiles pour des éléments tels que des pages de la Toile ou des cartes commerciales, ou même pour du personnel technique de soutien/aide qui veut envoyer des messages aux utilisateurs mais ne peut trouver lui-même le certificat. Ces URI peuvent aussi être utilisés pour appliquer des mesures de confidentialité quand on distribue les certificats en perturbant la clé de recherche d'une façon connue seulement du magasin de certificats/clés publiques, ou au magasin de certificats et aux utilisateurs (en d'autres termes, en convertissant l'URI en une capacité). Par exemple, un utilisateur avec un certificat qui vient d'être produit pourrait recevoir pour instruction d'aller le chercher avec une clé de "x-encrCertHash=...", qui est déchiffré par le magasin de certificats pour aller chercher le certificat approprié, en s'assurant que seul le possesseur du certificat peut aller chercher son certificat immédiatement après sa production. De même, une organisation qui ne veut pas rendre ses certificats disponibles à une interrogation publique peut exiger un MAC sur les clés de recherche (par exemple, "x-macCertHash=...") pour s'assurer que seuls des utilisateurs autorisés peuvent chercher les certificats (bien qu'un moyen plus logique pour le contrôle d'accès, si un vrai serveur de la Toile est utilisé pour accéder au magasin, serait évidemment le niveau HTTP).

Les types d'interrogation ont été spécifiquement choisis pour être non simplement une interface HTTP à LDAP mais un mécanisme général de restitution qui permette des mécanismes arbitraires de mémorisation de certificats/clés publiques (avec un biais vers de simples mémorisations {clé,valeur}, qui sont déployées presque universellement, avec ISAM, Berkeley DB, ou un RDBMS) pour être employés comme extrémités. La présente spécification a été écrite délibérément pour être technologiquement neutre, permettant à tout mécanisme de recherche de {clé,valeur} d'être utilisé. Il n'importe pas qu'on choisisse d'avoir des chimpanzés entraînés à chercher des certificats dans des tableaux tant que la méthode peut fournir la réponse correcte avec une raisonnable efficacité.

Les magasins de certificats/clés publiques et de CRL ont des URI séparés parce qu'ils peuvent être mis en œuvre en utilisant des mécanismes différents. Un magasin de certificats contient normalement de grands nombres de petits éléments, tandis qu'une mémorisation de CRL contient un très petit nombre d'éléments potentiellement grands. En fournissant des URI indépendants, il est possible de mettre en œuvre les deux magasins en utilisant des mécanismes taillés sur mesure pour les données qu'ils contiennent.

PGP combine les informations de clé et de révocation en un seul objet de données de sorte qu'il est possible de retourner à la fois les clés publiques et les informations de révocation à partir du même URI. Si des serveurs distincts de clé et de révocation sont disponibles, ils peuvent fournir un léger gain de performances car aller chercher les informations de révocation n'exige pas d'aller chercher la clé à laquelle elles s'appliquent. Si des serveurs séparés ne sont pas disponibles, un seul serveur peut être utilisé pour satisfaire les deux types d'interrogations avec une légère perte de performances, car aller chercher les informations de révocation va aussi aller chercher les données de clé publique associées aux données de révocation.

2.5.4 Réponses

L'interdiction de formes de codage exotiques reflète le fait que la plupart des clients (et beaucoup de serveurs, en particulier pour les appareils incorporés) ne sont pas des navigateurs généraux de la Toile ou des serveurs capables de traiter une gamme arbitraire de formes et types de codages, mais simplement des moteurs HTTP de base rattachés à des applications de gestion de clé. En d'autres termes, l'interface HTTP est un ajout rudimentaire à l'application de gestion de clé, plutôt que la gestion de clé n'est un ajout à un client ou serveur général de la Toile. Éliminer les choix inutiles simplifie la tâche de la mise en œuvre et réduit la taille et la complexité du code, avec une diminution de la probabilité de survenance de problème de sécurité découlant de l'accroissement de complexité.

L'utilisation d'un en-tête "Accept-encoding: identity" aurait le même effet qu'interdire tous codages supplémentaires et peut bien sûr être utile car le paragraphe 14.3 de la [RFC2616] indique que l'absence de cet en-tête peut être considérée comme signifiant que tout codage est permis. Cependant, cela gonfle inutilement l'en-tête HTTP d'une manière qui peut affecter les performances (voir au paragraphe 2.5.5) tandis qu'établir une exigence que la réponse soit retournée sans aucune décoration supplémentaire évite le besoin de le spécifier dans chaque demande. Les mises en œuvre devraient donc omettre l'en-tête Accept-encoding entièrement ou si il doit être inclus, inclure "identity" ou le caractère générique "*" comme type de

codage de contenu accepté.

L'utilisation de codage tronqué est donnée comme un "NE DEVRAIT PAS" plutôt qu'un "NE DOIT PAS" parce que sa prise en charge est exigée par la [RFC2616]. Néanmoins, cette forme de codage est fortement déconseillée, car les quantités de données transférées (1-2 kB) la rend entièrement inutile, et la prise en charge de cette forme de codage est vulnérable à diverses erreurs de mise en œuvre, dont certaines peuvent affecter la sécurité. Cependant, les développeurs devraient être conscients que de nombreuses versions du serveur Apache de la Toile va inutilement utiliser le codage tronqué en retournant les réponses. Bien qu'il soit préférable de faire de cela un "NE DOIT PAS", cela rendrait les clients qui le rejettent incompatibles avec le serveur le plus largement utilisé au monde sur la Toile. Pour cette raison, la prise en charge du codage tronqué est fortement déconseillée mais est néanmoins permise. Les clients qui choisissent de ne pas le prendre en charge devraient être conscients qu'ils peuvent avoir des problèmes de communication avec les magasins de certificats HTTP fondés sur Apache.

Plusieurs réponses sont retournées comme multiparties/mixtes plutôt que comme une SEQUENCE DE certificats ASN.1 ou une chaîne de certificats PKCS n° 7/CMS (données signées dégénérées contenant seulement des certificats) parce que c'est plus direct à mettre en œuvre avec des outils standard à capacité d'accès à la Toile. Un avantage supplémentaire est que cela ne restreint pas le mécanisme d'accès aux données fondées sur DER, lui permettant d'être étendu aux autres types de certificat, comme XML, PGP, et SPKI.

2.5.5 Questions de performances

Lorsque de forts débits/performances en charge sont un problème critique, une base de données de mémoire principale qui agit comme une forme d'antémémoire de contenu peut être interposée entre la base de données sur disque et l'interface HTTP [Garcia-Molina]. Une base de données de mémoire principale fournit la même fonctionnalité qu'une base de données sur disque et est pleinement transparente à l'extrémité frontale HTTP, mais offre des facilités de gestion de mémoire tampon et de restitution optimisées pour les données résidant en mémoire. Lorsque plus d'adaptabilité est requise, le système de mise en antémémoire de contenu peut être mis en œuvre comme une grappe de bases de données de mémoire principale [Ji].

Diverses considérations d'efficacité du réseau doivent être prises en compte lors de la mise en œuvre de ce mécanisme de distribution de certificats/clés publiques. Par exemple, une mise en œuvre simpliste qui effectue deux écritures (l'en-tête HTTP et le certificat, écrits séparément) suivies par une lecture, va mal interagir avec l'accusé de réception retardé de TCP et le démarrage lent. Cela se produit parce que le MSS TCP est normalement de 1460 octets sur un LAN (Ethernet) ou 512/536 octets sur un WAN, alors que les en-têtes HTTP font ~200-300 octets, beaucoup moins que le MSS. Quand un message HTTP est envoyé d'abord, la fenêtre d'encombrement TCP commence à un segment, avec le démarrage lent de TCP qui double sa taille pour chaque ACK. Envoyer les en-têtes séparément va envoyer un segment court et un second segment de taille MSS, sur quoi la pile TCP va attendre l'accusé de réception du répondant avant de continuer. Le répondant obtient les deux segments, puis retarde son ACK de 200 ms dans l'espoir de le faire porter sur des données de réponse, qui ne sont jamais envoyées, car il attend toujours le reste du corps HTTP en provenance de l'initiateur. Il en résulte qu'il y a un retard de 200 ms (+ le RTT assorti) pour chaque message envoyé.

Diverses autres considérations doivent être prises en compte pour fournir un maximum d'efficacité. Elles sont couvertes en profondeur dans [Spero], [Heidemann], [Nielsen]. De plus, des modifications du comportement de TCP, comme l'utilisation de fenêtres initiales de 4K [RFC3390] (conçues pour réduire les petits temps de transfert HTTP à un seul RTT) devraient aussi améliorer certains de ces problèmes.

Un règle d'approximation des performances optimales est de combiner l'en-tête HTTP et la charge utile de données en une seule écriture (toute mise en œuvre raisonnable de HTTP va le faire de toutes façons, grâce à l'expérience considérable qui existe pour le réglage des performances des serveurs HTTP) et de garder les en-têtes HTTP à leur minimum pour essayer de faire tenir les données dans le TCP MSS. Par exemple, comme ce protocole n'implique pas de navigateur de la Toile, il n'est pas besoin d'inclure divers en-têtes communs relatifs au navigateur comme ceux qui détaillent les versions de logiciel ou les langues acceptables.

2.5.6 Divers

L'interface spécifiée dans ce document est de type basique en lecture seule qui va être utilisé par la majorité des clients. Le traitement des mises à jour (insertion et suppression) est un problème complexe qui implique des questions de technologie (la diversité des champs utilisés pour l'indexation et la restitution d'informations doit être spécifiée d'une façon neutre quant à la technologie, ou le magasin de certificats doit effectuer sa propre analyse des éléments ajoutés, passant d'un mécanisme

de recherche de clé=valeur presque universel à un système de traitement complet de clé publique/certificat) et des questions de politique (qui peut effectuer les mises à jour du magasin de certificats, et dans quelles conditions ?). À cause de cette complexité, les détails de tout mécanisme de mise à jour potentielle est laissé à la configuration locale, bien qu'ils pourraient être couverts par un futur document si il y a une demande suffisante.

Des questions ont été soulevées quant à l'utilisation de HTTP comme sous strate [RFC3205]. Le mécanisme décrit ici, qui met en œuvre un protocole direct de demande/réponse avec la même sémantique que les demandes HTTP traditionnelles, n'est pas affecté par ces questions. Précisément, il ne met en œuvre aucune forme de mécanisme RPC complexe, n'exige pas de mesures de sécurité HTTP, n'est pas affecté par les pare-feu (car il utilise seulement un GET HTTP de base plutôt que de mettre en couche un nouveau protocole par dessus HTTP) et a des types bien définis de supports MIME spécifiés dans des documents normatifs. À ce titre, les soucis exprimés dans la [RFC3205] ne s'appliquent pas ici. De plus, bien qu'un certain nombre de serveurs ne prennent toujours pas pleinement en charge certaines des caractéristiques les plus avancées de HTTP 1.1 [Krishna], le sous ensemble minimal utilisé ici est bien pris en charge par la majorité des serveurs et mises en œuvre de HTTP.

Ce mécanisme d'accès est similaire au protocole HKP de PGP [HKP] ; cependant, ce dernier est presque entièrement non documenté et exige que les mises en œuvre reviennent sur l'ingénierie d'autres mises en œuvre. À cause de ce manque de normalisation, aucune tentative n'a été faite pour assurer l'interopérabilité ou la compatibilité avec les serveurs fondés sur HKP, bien que les développeurs de PGP aient fourni beaucoup de précieux apports au présent document. Un des avantages qu'apporte HKP est l'extensive expérience de mise en œuvre, qui indique que c'est une solution très envisageable au problème d'un mécanisme simple de restitution de certificat/clé publique. Les serveurs HKP ont été mis en œuvre en utilisant des fichiers plats, Berkeley DB, et diverses bases de données, comme Postgres et MySQL.

2.6 Exemples

Pour convertir le DN sujet C=NZ, O=... CN=Fred Dagg en une clé de recherche :

Hacher le DN, en forme codée en DER dans laquelle il apparaît dans le certificat, pour obtenir :

96 4C 70 C4 1E C9 08 E5 CA 45 25 10 D6 C8 28 3A 1A C1 DF E2

Coder cela en base-64 pour obtenir : lkxwxB7JCOXKRSUQ1sgoOhrB3+I
(Noter l'absence de bourrage de '=' en queue.) C'est la clé de recherche à utiliser dans l'interrogation d'URI.

Pour aller chercher tous les certificats utiles pour envoyer un message chiffré à foo@exemple.com :

```
GET /search.cgi?email=foo%40exemple.com HTTP/1.1
```

(Pour faire simple, l'en-tête supplémentaire Host: header exigé par la [RFC2616] est omis ici et dans les exemples suivants.) Dans ce cas, "/search.cgi" est la portion abs_path de l'URI d'interrogation, et la demande est soumise au serveur situé à la portion net_loc de l'URI d'interrogation. Noter le codage du symbole "@"' selon la [RFC2854]. Les en-têtes exigés restants, comme l'en-tête "Host" exigé par HTTP 1.1, ont été omis dans un souci de clarté.

Pour aller chercher le certificat de la CA qui a produit le certificat du message électronique :

```
<Convertir le DN du producteur en clé de recherche>
    GET /search.cgi?sHash=<clé de recherche> HTTP/1.1
```

Autrement, si le chaînage est par identifiant de clé :

```
<Extraire le keyIdentifier de authorityKeyIdentifier>
    GET /search.cgi?sKIDHash=<clé de recherche> HTTP/1.1
```

Pour aller chercher les autres certificats appartenant au même utilisateur que le certificat de message électronique :

```
<Convertir le DN sujet en clé de recherche>
    GET /search.cgi?sHash=<clé de recherche> HTTP/1.1
```

Pour aller chercher la CRL pour le certificat :

<Convertir le DN du producteur en une clé de recherche>
GET /search.cgi?iHash=<clé de recherche > HTTP/1.1

Noter que comme le différenciateur est l'URI de base, les deux interrogations ci-dessus apparaissent identiques (car l'URI de base n'est pas montré) mais en fait elles sont distinctes.

Pour restituer une clé en utilisant les méthodes XML, le <NomDeClé> (qui contient l'identifiant de chaîne pour la clé) utilisé avec le hachage de DN sujet DN ci-dessus, serait :

< NomDeClé KeyID="sHash">lkxwxB7JCOXKRSUQ1sgoOhrB3+I</NomDeClé>.

3. Localisation des magasins de certificats HTTP

Afin de localiser les serveurs d'où les certificats peuvent être restitués, les parties peuvent employer une ou plusieurs des stratégies suivantes :
- Informations contenues dans le certificat
- Utilisation de SRV DNS
- Utilisation d'un site "bien connu"
- Configuration manuelle du logiciel client

L'intention des diverses options fournies ici est de rendre l'accès au magasin de certificats aussi transparent que possible, en exigeant seulement la configuration manuelle par l'utilisateur en dernier ressort.

3.1 Informations dans le certificat

Afin de porter un accès de point d'informations bien connu aux parties, les CA DEVRAIENT utiliser l'extension SubjectInfoAccess (SIA) et AuthorityInfoAccess (AIA) [RFC3280] dans les certificats. La valeur de l'OID pour la méthode d'accès est un des suivants :

IDENTIFIANT D'OBJET id-ad-http-certs ::= { id-ad 6 }
IDENTIFIANT D'OBJET id-ad-http-crls ::= { id-ad 7 }

où :

IDENTIFIANT D'OBJET id-ad ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) 48 }

La localisation d'accès correspondante est l'URI d'interrogation. L'utilisation de cette facilité donne à une CA une localisation standard pratique pour indiquer où plus de certificats peuvent être trouvés, par exemple, pour les besoins de la construction d'un chemin de certification. Noter que cela ne signifie pas que la fourniture de l'accès aux services du magasin de certificats est limité seulement aux CA.

3.2 Utilisation des SRV DNS

Le SRV du DNS est une facilité pour spécifier la localisation du ou des serveurs pour un certain protocole et domaine [RFC2782]. Pour l'interface de magasin de certificats, le nom symbolique de SRV du DNS pour l'interface de magasin de certificats DEVRA être "certificates". Le nom pour l'interface de mémorisation de CRL DEVRA être "crls". Le nom pour le magasin PGP de clé publique DEVRA être "pgpkeys". Le nom pour le magasin de révocation PGP DEVRA être "pgprevocations". Le traitement de facilités supplémentaires de SRV du DNS, comme les champs de priorité et de pondération, est selon la [RFC2782].

3.2.1 Exemple

Si une CA avec le domaine exemple.com devrait rendre disponibles ses certificats via une interface HTTP de magasin de certificats, les détails du serveur pourraient être obtenus par une recherche sur :

_certificates._tcp.exemple.com

et

_crls._tcp.exemple.com

Cela va retourner le ou les serveurs et accès pour le service comme spécifié dans la [RFC2782].

3.3 Utilisation de localisation "bien connue"

Si aucune autre information de localisation n'est disponible, l'interface de magasin de certificats peut être localisée à une localisation "bien connue" construite à partir du nom de domaine du fournisseur de service. Dans le cas usuel, l'URI est construit en ajoutant le type d'informations à restituer ("certificates.", "crls.", "pgpkeys.", ou "pgprevocations.") devant le nom de domaine pour obtenir la portion net_loc de l'URI, et en ajoutant une portion fixée abs_path "search.cgi". La forme d'URI de la localisation "bien connue" est donc :

certificates.<nom_de_domaine>/search.cgi
crls.<nom_de_domaine>/search.cgi
pgpkeys.<nom_de_domaine>/search.cgi
pgprevocations.<nom_de_domaine>/search.cgi

Les fournisseurs de service de magasin de certificats DEVRAIENT utiliser ces URI de préférence aux autres solutions. Noter que l'utilisation de "search.cgi" n'implique pas l'utilisation de scripts CGI [RFC3875]. Ce serait plutôt l'exception que la règle, car cela conduirait à une mise en œuvre assez inefficace ; cela fournit simplement une solution de remplacement de mise en œuvre possible (et relativement simple à établir) (voir les raisons pour plus de détails).

Un second cas se produit quand le service d'accès aux certificats est fourni par des appareils incorporés à capacité d'accès à la Toile, comme des appareils de branchement direct universel (UPNP, *Universal Plug and Play*) [UPNP]. Ces appareils ont un seul net_loc fixe (soit une adresse IP, soit un nom DNS) et rendent les services disponibles via une interface HTTP. Dans ce cas, l'URI est construit en ajoutant une portion abs_path fixe "certificates/search.cgi" pour les certificats, "crls/search.cgi" pour les CRL, "pgpkeys/search.cgi" pour les clés publiques PGP, et "pgprevocations/search.cgi" pour les informations PGP de révocation à la net_loc. La forme d'URI de la localisation "bien connue" est donc :

<net_loc>/certificates/search.cgi
<net_loc>/crls/search.cgi
<net_loc>/pgpkeys/search.cgi
<net_loc>/pgprevocations/search.cgi

Si l'accès aux certificats décrit dans le présent document est mis en œuvre par l'appareil, il DEVRAIT alors utiliser ces URI de préférence à d'autres solutions (voir les raisons pour les détails de cette exigence).

3.3.1 Exemples

Si une CA avec le domaine exemple.com devait rendre ses certificats disponibles via une interface HTTP de magasin de certificats, les URI d'interrogation "bien connus" pour les certificats et les CRL seraient :

http://certificates.exemple.com/search.cgi
http://crls.exemple.com/search.cgi

Un contrôleur automatique résident avec l'adresse IP 192.0.2.1 (un point de contrôle dans la terminologie UPnP) rendrait disponibles des certificats pour des appareils comme des contrôleurs HVAC, des contrôleurs d'éclairage et d'application, et des appareils de détection d'incendie et d'intrusion physique comme :

http://192.0.2.1/certificates/search.cgi
http://192.0.2.1/crls/search.cgi

Un serveur d'impression avec le nom DNS de "printspooler" rendrait disponibles des certificats pour des imprimantes sur la Toile avec lesquelles il communique comme :

http://printspooler/certificates/search.cgi
http://printspooler/crls/search.cgi

3.4 Configuration manuelle du logiciel de client

La localisation d'accès pour le magasin de certificats/clés publiques/CRL HTML PEUT être configuré localement chez le client. Cela peut être utilisé si aucune autre information n'est disponible, ou si il est nécessaire d'outrepasser d'autres informations.

3.5 Notes de mise en œuvre et raisons

Ces paragraphes d'informations documentent les raisons derrière la conception de la Section 3 et donnent des lignes directrice pour les mises en œuvre.

3.5.1 SRV DNS

La solution optimale au problème de la localisation de service serait le SRV du DNS. Malheureusement, le système d'exploitation utilisé par le groupe d'utilisateur qui a le plus besoin de ce type de traitement ne prend rien en charge au delà des recherches d'adresse DNS les plus basiques, rendant impossible l'utilisation du SRV DNS avec autre chose que les très récents systèmes Win2K et XP. Pour rendre les choses encore plus amusantes, plusieurs des noms de fonction et certains des paramètres de fonction ont changé à plusieurs reprises durant la phase de développement de Win2K, et le comportement des portions de l'API de prises Windows a changé de façon non documentée. Cela conduit à une situation malheureuse dans laquelle un administrateur de système Unix peut utiliser le SRV DNS pour éviter d'avoir à traiter des problèmes de configuration techniques, mais un utilisateur de Windows'95 ne le peut pas. À cause de ces problèmes, une solution de remplacement au SRV DNS est fournie pour les situations où il n'est pas possible de l'utiliser.

Le SRV ou l'option de localisation "bien connue" peut fréquemment être automatiquement déduite par le logiciel d'utilisateur à partir de paramètres connus. Par exemple, si l'adresse de messagerie du receveur est @exemple.com, le logiciel de l'utilisateur va interroger _certificates._tcp.exemple.com ou aller à certificates.exemple.com et demander le certificat. De plus, le logiciel d'utilisateur peut tenir une liste des sources de certificat connues de la façon dont les listes de CA connues sont tenues par les navigateurs. La mention spécifique de la prise en charge de la redirection à la Section 2 souligne que de nombreux sites vont externaliser la tâche de mémorisation de certificats. Au pire, tout ce qui va être exigé est l'ajout d'une seule page statique de la Toile pointant sur le serveur réel. Des solutions de remplacement comme les enregistrements de ressource CNAME du DNS sont aussi possibles mais peuvent n'être pas faciles à établir comme redirections HTTP (les politiques d'entreprise tendent à être plus souples à l'égard des contenus de pages de la Toile que ne le seraient la modification des configurations du DNS).

3.5.2 Localisations "bien connues"

L'URI de localisation "bien connu" est conçu pour rendre les options d'hébergement aussi souples que possible. Localiser le service à www.<nom de domaine> va généralement exiger qu'il soit traité par le serveur principal de la Toile du fournisseur, tandis qu'utiliser un URI de serveur distinct lui permet d'être traité comme désiré par le fournisseur. Bien qu'on ne puisse douter que des serveurs mettent en œuvre l'interface qui utilise des scripts Apache et Perl, une mise en œuvre plus logique va consister en une simple interface réseau à un mécanisme de recherche de clé-et-valeur, comme la Berkeley DB. La forme d'URI présenté au paragraphe 3.3 permet une souplesse maximale, car elle fonctionne aussi bien avec les scripts de serveurs de la Toile/CGI que des extrémités frontales de réseau de serveur non sur la Toile pour les magasins de certificats.

3.5.3 Informations dans le certificat

Les mises en œuvre qui exigent l'utilisation de localisations, d'accès, non standard, ou de HTTPS plutôt que HTTP en combinaison avec des localisations "bien connues" devraient utiliser une redirection HTTP à la localisation "bien connue" pour pointer sur la localisation non standard. Par exemple, si le pôle d'impression du paragraphe 3.3 utilise un serveur protégé par SSL nommé printspooler-serveur avec une portion abs_path de cert_access, il va utiliser une redirection HTTP 302 pour https://printspooler-serveur/cert_access. Cela combine la capacité de connexion directe *(plug-and-play)* des localisations "bien connues" avec la capacité d'utiliser des localisations et accès non standard.

Les extensions SIA et AIA sont utilisées pour indiquer la localisation de l'interface de mémorisation de CRL plutôt que l'extension CRLDistributionPoint (CRLDP, *point de distribution de CRL*) car les deux effectuent des fonctions entièrement différentes. Un CRLDP contient "un pointeur sur la CRL en cours", localisation fixe contenant une CRL pour le certificat en cours, tandis que l'extension SIA/AIA indique "comment accéder aux informations et services de CA pour le sujet/producteur du certificat dans lequel apparaît l'extension", dans ce cas, l'interface de mémorisation de CRL qui fournit

les CRL pour tout certificat produit par la CA. De plus, CRLDP "associe" d'autres informations d'attribut avec une interrogation qui est incompatible avec les simples mécanismes d'interrogation présentés dans ce document.

Un seul serveur peut être utilisé pour traiter les interrogations CRLDP et AIA/SIA pourvu que la forme CRLDP utilise un URI HTTP. Comme CRLDP pointe sur une seule localisation statique pour une CRL, une interrogation peut être pré construite et mémorisée dans l'extension CRLDP. Le logiciel qui utilise le CRLDP va restituer la seule CRL qui s'applique au certificat venant du serveur, et le logiciel qui utilise AIA/SIA peut restituer toute CRL du serveur. Des URI pré construits similaires peuvent aussi être utiles dans d'autres circonstances (par exemple, pour des liaisons sur des pages de la Toile) pour les placer dans les localisations appropriées comme le issuerAltName, ou même pour le personnel de support technique/bureau d'assistance à la messagerie pour les utilisateurs qui ne peuvent pas trouver eux-mêmes le certificat, comme décrit au paragraphe 2.5. L'URI certstore résultant, quand il est cliqué par l'usager, va directement accéder au certificat quand c'est utilisé en conjonction avec une application à capacité de certificat, comme un navigateur ou un programme de messagerie.

3.5.4. Divers

Les appareils qui ont la capacité d'accéder à la Toile (ou, plus exactement, qui ont la capacité HTTP) sont destinés à être à accès direct à l'Internet, avec une configuration d'utilisateur nécessaire minimale (ou pas). L'URI "bien connu" permet à tout appareil connu (par exemple, découvert via le protocole simple de découverte de service (SSDP, *Simple Service Discovery Protocol*) de UPnP d'être interrogé sur des certificats sans exiger d'autre configuration de la part de l'utilisateur. Noter qu'en pratique aucun appareil incorporé ne va jamais utiliser l'adresse donnée dans l'exemple (l'adresse standard de fait pour les appareils à accès Internet incorporé est 192.168.1.x et non 192.0.2.x) ; cependant, la politique de l'IETF exige l'utilisation de cette non adresse pour les exemples.

Des protocoles comme UPnP ont leurs propres moyens de disséminer les informations d'appareil et de protocole. Par exemple, UPnP utilise SOAP, qui fournit une action GetPublicKeys pour tirer les clés d'appareils et une action PresentKeys pour pousser les clés de point de contrôle. Le texte du paragraphe 3.3 n'est pas destiné à impliquer que ce document se substitue au mécanisme UPnP existant, mais simplement que, si un appareil met en œuvre le mécanisme décrit ici, il devrait utiliser le schéma de dénomination du paragraphe 3.3 plutôt que des noms arbitraires.

4. Considérations sur la sécurité

Les mandataires d'antémémoire HTTP sont courants sur l'Internet, et certains mandataires peuvent ne pas vérifier correctement la dernière version d'un objet. La [RFC2616] spécifie que les réponses aux URL d'interrogation ne devraient pas être mis en antémémoire, et la plupart des mandataires et serveurs mettent en œuvre correctement le mécanisme "Cache-Control: no-cache" qui peut être utilisé pour outrepasser la mise en antémémoire ("Pragma: no-cache" pour HTTP 1.0). Cependant, dans les rares instances dans lesquelles une demande HTTP pour un certificat ou une CRL passe à travers un mandataire mal configuré ou autrement défaillant, le mandataire peut retourner une réponse obsolète.

Il faut veiller à s'assurer que seules des interrogations valides sont fournies à l'extrémité utilisée pour restituer des certificats. Permettre à des attaquants de soumettre des interrogations arbitraires peut leur permettre de manipuler le magasin de certificats de façon inattendue si l'extrémité essaye d'interpréter le contenu de l'interrogation. Par exemple, si un magasin de certificats est mis en œuvre en utilisant un RDBMS pour lequel l'application appelante assemble une chaîne SQL complète pour effectuer l'interrogation, et si l'interrogation SQL est construite comme "SELECT certificate FROM certificates WHERE iHash = " + <clé de recherche>, et si <clé de recherche> est réglé à "X;DELETE FROM certificates", le résultat de l'interrogation va être assez différent de ce qui était attendu par l'administrateur du magasin de certificats. La même chose s'applique aux interrogations par nom et adresse de messagerie. Même une interrogation en lecture seule peut être problématique ; par exemple, régler <clé de recherche> à "UNION SELECT password FROM master.sysxlogins" va faire la liste de tous les mots de passe dans une base de données de serveur SQL (dans un format facile à déchiffrer) si l'utilisateur est sous le compte "sa" (DBA). Une vérification directe de bonne santé des interrogations peut n'être pas suffisante pour empêcher toutes les attaques ; par exemple, un filtre qui retire la chaîne d'interrogation SQL "DELETE" peut être outrepassé en soumettant la chaîne incorporée dans une autre instance de la chaîne. Retirer "DELETE" de "DELDELETEETE" laisse le "DELETE" extérieur en place. Abuser la troncature de chaînes très longues par des filtres peut aussi être utilisé comme moyen d'attaque, dans laquelle l'attaquant s'assure que la troncature se produit au milieu d'une séquence d'échappement, outrepassant le filtrage. L'utilisation des interrogations paramétrées (souvent appelées bouche trous) qui ne sont pas vulnérables à l'injection SQL devrait être utilisée pour éviter ces attaques.

De plus, comme certaines données d'interrogation peuvent être codées/décodées avant d'être envoyées à l'extrémité, les applications devraient vérifiée la validité des données de la forme codée et décodée. Un moyen simple pour éviter ces

problèmes est d'utiliser des commandes paramétrées plutôt que les chaînes SQL assemblées manuellement pour les utiliser dans les interrogations (ceci est aussi plus efficace pour la plupart des interfaces de base de données). L'utilisation de commandes paramétrées signifie que la valeur d'interrogation n'est jamais présente dans une position où elle pourrait être interprétée comme une portion de la commande d'interrogation.

À côté du filtrage d'interrogations, l'extrémité arrière devrait être configurée à désactiver toute forme d'accès de mise à jours via l'interface de la Toile. Pour Berkeley DB, cette restriction peut être imposée en ouvrant le magasin de certificats en mode lecture seule à partir de l'interface d'accès Internet. Pour les bases de données relationnelles, cela peut être imposé par le mécanisme SQL GRANT/REVOKE, par exemple, "REVOKE ALL ON certificates FROM webuser. GRANT SELECT ON certificates TO webuser" va permettre l'accès en lecture seule de la sorte appropriée pour l'interface d'accès Internet. Des mesures de sécurité spécifiques du serveur peuvent aussi être employées ; par exemple, le serveur SQL fournit le compte db_datareader incorporé qui permet seulement l'accès en lecture aux tableaux (mais voir la note ci-dessus sur ce qui peut être fait même avec un accès en lecture seule) et la capacité de faire fonctionner le serveur sous un compte dédié à faible privilège (une caractéristique standard des systèmes Unix).

Le mécanisme décrit dans ce document n'est pas destiné à fonctionner comme un répertoire/base de données de confiance. En particulier, les utilisateurs ne devraient pas supposer que juste parce qu'ils sont allés chercher une clé publique ou un certificat auprès d'une entité qui prétend être X, X fait une déclaration sur la véracité de la clé publique ou du certificat. L'utilisation d'une représentation signée des éléments mémorisés supprime le besoin de dépendre du magasin de certificats pour tout service de sécurité autre que la disponibilité. Bien qu'il soit possible de mettre en œuvre un répertoire/base de données de confiance en utilisant HTTPS ou une autre forme de liaison sécurisé/de confiance, c'est un problème de politique/configuration locale, et en l'absence de telles mesures de sécurité supplémentaires, les utilisateurs devraient appliquer les niveaux appropriés de vérification à toute clé ou certificat qu'ils vont chercher avant de l'utiliser.

5. Considérations relatives à l'IANA

Aucune action de l'IANA n'est nécessaire. Les types de méthode d'accès AIA/SIA sont identifiés par des identifiants d'objet (OID) à partir d'un arc géré par le groupe de travail PKIX. Si des méthodes d'accès supplémentaires devaient être introduites (par exemple, pour des certificats d'attribut ou des types de certificat non X.509) les promoteurs de telles méthodes d'accès sont supposés allouer les OID nécessaires à partir de leurs propres arcs.

6. Remerciements

Anders Rundgren, Blake Ramsdell, Jeff Jacoby, David Shaw, et les membres du groupe de travail ietf-pkix ont fourni d'utiles apports et retours sur le présent document.

7. Références

7.1. Références normatives

[FIPS180] Federal Information Processing Standards Publication (FIPS PUB) 180-1, "Secure Hash Standard", 17 avril 1995.

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. *(MàJ par* RFC8174)

[RFC2440] J. Callas, L. Donnerhacke, H. Finney et R. Thayer, "Format de message OpenPGP", novembre 1998. *(Obs. voir* 4880)

[RFC2585] R. Housley et P. Hoffman, "Protocoles de fonctionnement de l'infrastructure de clé publique X.509 pour l'Internet : FTP et HTTP", mai 1999. *(P.S.)*

[RFC2616] R. Fielding et autres, "Protocole de transfert hypertexte -- HTTP/1.1", juin 1999. *(D.S., MàJ par* 2817, 6585)

[RFC2782] A. Gulbrandsen, P. Vixie et L. Esibov, "Enregistrement de ressource DNS pour la spécification de la localisation des services (DNS SRV)", février 2000.

[RFC2854] D. Connelly et L. Masinter, "Type de support 'text/html'", juin 2000. *(Information)*

[RFC3156] M. Elkins et autres, "Sécurité MIME avec OpenPGP", août 2001.*(P.S.)*

[RFC3275] D. Eastlake 3rd , J. Reagle, D. Solo, "Syntaxe et traitement de signature en langage de balisage extensible (XML)", mars 2002. *(D.S.)*

[RFC3280] R. Housley, W. Polk, W. Ford et D. Solo, "Profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", avril 2002. *(Obsolète, voir* RFC5280)

[RFC3852] R. Housley, "Syntaxe de message cryptographique (CMS)", juillet 2004. *(Obsolète, voir la* RFC5652)

7.2 Références pour information

[Birkholz] Erik Birkholz et al, "Special Ops: Host et Network Security for Microsoft, Unix, et Oracle", Syngress Publishing, novembre 2002.

[Garcia-Molina] Hector Garcia-Molina and Kenneth Salem, "Main Memory Database Systems", IEEE Transactions on Knowledge et Data Engineering, Vol.4, No.6 (décembre 1992), p.509.

[Gutmann] P. Gutmann, "A Reliable, Scalable General-purpose Certificate Store", Proceedings of the 16th Annual Computer Security Applications Conference, décembre 2000.

[Heidemann] J. Heidemann, "Performance Interactions Between P-HTTP et TCP Implementations", ACM Computer Communications Review, avril 1997.

[HKP] Marc Horowitz, "A PGP Public Key Server", 2000, http://www.mit.edu/afs/net.mit.edu/project/pks/thesis/paper/thesis.html. On peut obtenir une vue plus complète et à jour de HKP à partir du code source d'une mise en œuvre OpenPGP libre comme GPG.

[Ji] Minwen Ji, "Affinity-based Management of Main Memory Database Clusters", ACM Transactions on Internet Technology, Vol.2, n° 4 (novembre 2002), p.307.

[Krishna] Balachander Krishnamurthy et Martin Arlitt, "PRO-COW: Protocol Compliance on the Web - A Longitudinal Survey", Proceedings of the 3rd Usenix Symposium on Internet Technologies et Systems (USITS'01), mars 2001, p.109.

[Nielsen] H.Nielsen, J.Gettys, A.Baird-Smith, E.Prud'hommeaux, H.Wium Lie, et C.Lilley, "Network Performance Effects of HTTP/1.1, CSS1, et PNG", 24 juin 1997, http://www.w3.org/Protocols/HTTP/Performance/Pipeline.html

[PKCS11] PKCS #11 Cryptographic Token Interface Standard, RSA Laboratories, décembre 1999.

[PKCS15] PKCS #15 Cryptographic Token Information Syntax Standard, RSA Laboratories, juin 2000.

[RFC3205] K. Moore, "Sur l'utilisation de HTTP comme sous strate", février 2002. (BCP0056)

[RFC3390] M. Allman, S. Floyd, C. Partridge, "Augmentation de la fenêtre initiale de TCP", octobre 2002. *(P.S.)*

[RFC3875] D. Robinson, K. Coar, "Interface de passerelle commune (CGI) version 1.1", octobre 2004. *(Information)*

[Spero] S.Spero, "Analysis of HTTP Performance Problems", juillet 1994, http://www.w3.org/Protocols/HTTP/1.0/HTTPPerformance.html .

[UPNP] "Universal Plug et Play Device Architecture, Version 1.0", UPnP Forum, 8 juin 2000.

Adresse de l'auteur

Peter Gutmann
University of Auckland
Private Bag 92019
Auckland, New Zealand

mél : pgut001@cs.auckland.ac.nz

Déclaration complète de droits de reproduction

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