La Sécurité sur le Web

Marc Barret
INRIA Rhône-Alpes

Marc.Barret@inria.fr

De quoi se protéger?

Le fait de mettre en service un serveur Web implique d'ouvrir sa machine à la communauté Internet, et d'accepter que nimporte quel membre de cette communauté y consomme les ressources fournies par le serveur. L'administrateur du serveur veut cependant être sûr qu'un individu mal intentionné ne puisse consommer plus de ressources que prévu, détruire des informations ou accéder à des informations qu'il veut garder secrètes. C'est par le terme de **piratage** qu'on nomme ce danger.

De même, un consommateur de ressources veut être sûr que l'information reçue d'un serveur Web ne contienne pas de **virus**, et un administrateur de serveur ne veut pas qu'un consommateur puisse y installer un virus.

Avec le succés du Web, et son nombre croissant d'utilisateurs, de nouvelles applications ont vu le jour. Certaines d'entre elles, plus sensibles que celles pour lequel le Web a été conçu, ont créé de nouvelles contraintes, plus ou moins satisfaites par les structures du Web et de l'Internet:

- **contrôle d'accés**: seuls certains clients préalablement autorisés doivent pouvoir accéder à certaine information
- **authentification**: le client (serveur) veut être sur que le serveur (client) est bien qui il prétend être.
- **confidentialité**: les deux parties veulent être sûres que l'information échangée ne puisse être lue par une tierce personne.
- **intégrité**: être sur que les données ont été correctement transmises.

Le Piratage

Les exemples connus de piratage à travers le Web sont rares.

Piratage du serveur

L'administrateur d'un serveur Web peut définir le *User Id* sous lequel le serveur va s'exécuter. L'administrateur consciencieux attribuera donc au serveur Web un *User Id* sans privilèges particuliers (en général tel que `nobody`, et surtout pas `root` ...). Dans ce cas, le serveur Web, et donc le pirate qui l'utilise à mauvais escient, ne pourra pas faire plus de dommages que ce que l'y autorisent les droits de cet *User Id*.

Le protocole HTTP est trés simple, et ne prévoit que quelques requêtes du client vers le serveur: **GET**, **POST** et **PUT**. Les deux premières, implémentés par tous les types de serveur,

font renvoyer de l'information au client par le serveur. Cette information peut être soit le contenu d'un fichier hébergé sur le serveur, soit le résultat de l'exécution d'un programme, appelé dans ce cas une **passerelle CGI** (Common Gateway Interface). Dans ce cas, il est du ressort de l'administrateur de s'assurer que ces programmes, mis à la disposition de la communauté, ne présentent pas de danger pour l'intégrité du serveur. Il faut cependant garder à l'esprit que ces programmes vont s'exécuter avec les droits du *User Id* attribué au serveur par l'administrateur.

La requête **PUT** permet à un client d'écrire (et donc d'effacer) un document sur le serveur. Cette requête n'est pas implémentée sur la plupart des serveurs (le serveur du CERN est un des rares à le faire). Son utilisation est soumise à des contrôles d'accés.

Un mécanisme analogue à celui des passerelles CGI permet au serveur d'interpréter un document avant de le retourner au client, c'est la fonctionalité appelée **Server Side Includes**. Cette fonctionalité doit être activée explicitement par l'administrateur du serveur. Parmi les commandes interprétées peuvent se trouver des requêtes d'exécution de programmes. Là encore, il est du ressort de l'administrateur de s'assurer de l'innocence de ces programmes auxquels il donne un accés public via le Web.

Le piratage peut être facilité par l'exploitation de failles du programme serveur. Ainsi en Mars 1995, il est apparu qu'une bogue du serveur HTPPD du NCSA, version 1.3, permettait de faire exécuter par le programme serveur nimporte quelle commande. Les pirates potentiels devaient cependant être au courant de cette faille, et de la façon de l'exploiter.

Une façon de diminuer les risques de piratage sur un serveur Web pourrait être de l'exécuter dans un espace disque limité par la commande système Unix `chroot`. Outre le fait que cette fonctionalité n'existe pas sur les autres systèmes d'exploitation, son utilisation implique de recréer un *file system* sous la nouvelle racine du serveur, contenant les fichiers système dont il a besoin: fichiers *device*, librairies partagées, interpréteurs de commandes... De plus, les directories où les utilisateurs peuvent rendre leurs propres documents accessibles (en général appelés `$HOME/public_html`) sont ainsi invisibles du serveur.

Le rôle de l'administrateur du serveur Web apparaît donc comme prépondérant pour la sécurité.

Piratage du client

Le client Web peut également être sujet au piratage. Il est ainsi possible de configurer, au niveau des *helper applications* du client, un type de document par exemple appelé `application/x-csh` auquel on associe la commande `csh -f %`. Lorsque le client ainsi configuré reçoit un document de ce type, il va l'exécuter avec l'interpréteur de commande csh. Il ne reste alors plus qu'à souhaiter l'innocence du script envoyé par le serveur...

Le nouveau browser développé par Sun Microsystems, appelé **Hot Java**, permet de charger dynamiquement du code exécutable du serveur vers le client. Ces morceaux de code, appelés *applets*, vont donc s'exécuter sur le client. Bien qu''il semble ainsi facile de transmettre une applet contaminante, il ne semble pas que le problème se soit présenté à ce jour. Dans un souci de sécurité, le langage Java, dans lequel les applets sont écrites, n'offre pas de fonctions d'appels au système (lecture, écriture sur disque, exécution de comandes...)

Les Virus

Les virus se sont principalement manifestés sur des machines dont le système d'exploitation ne présente pas de notions de *User Id*, et de protections associées, c□'est à dire Mac OS et MS-Dos/Windows. Des virus aussi virulents et nuisibles que ceux qu'on a pu rencontrer sur ces systèmes ne se sont jamais manifestés, à notre connaissance, sur des systèmes Unix.

Nous ne connaissons pas d'exemples de propagation de virus utilisant le Web comme vecteur de contamination.

On imagine facilement que la requête **PUT**, utilisée à mauvais escient, pourrait à priori permettre l'installation d'un virus sur le serveur par un client.

Contrôle d'Accés et Authentification

Certaines applications peuvent nécessiter de restreindre l'accés à un serveur (ou à certains de ses documents). La plupart des serveurs HTTP existant fournissent cette fonctionalité à l'administrateur, sous la forme de *Listes de Contrôle d'Accés*, ou **ACL** en anglais pour *Access Control List*. Les critères d'autorisation peuvent être:

- nom d'utilisateur et mot de passe: l'administrateur créé des noms d'utilisateurs et mots de passe, et les transmet aux personnes autorisées. Pour accéder à un document d'accés ainsi controlé, le client reçoit du serveur l'information comme quoi le document est protégé par ce mécanisme. Le client demande alors à son utilisateur un nom d'utilisateur et un mot de passe, qu'il transmet au serveur. Si ceux ci sont corrects vis à vis de la base de données du serveur, le serveur retourne le document demandé. Il faut noter que le mot de passe est transmis uniquement codé avec le codage *uuencode*, que tout le monde peut décoder.

- nom d'utilisateur, mot de passe et appartenance à un groupe: l'administrateur crée des noms d'utilisateurs avec mots de passe, et des groupes consistant en une liste d'utilisateurs. Aprés avoir vérifié les validités du nom d'utilisateur et du mot de passe, le serveur vérifie l'appartenance de l'utilisateur à un groupe autorisé.

- appartenance de la machine client à un domaine ou à un réseau: l'administateur peut restreindre l'accés à des requêtes provenant de domaines, de réseaux ou d'adresses IP préalablement déterminés. Mais l'ordinateur émettant la requête peut être déja lui même piraté: c'est l'ordinateur qui est identifié, pas l'utilisateur. De plus, dans certaines conditions bien spécifiques, un pirate averti, équipé du matériel et logiciel adéquat, peut usurper une adresse IP qui n'est pas la sienne (*spoofing*, ou *mascarade*). Un autre inconvénient à ce type de contrôle est que si un client utilise un *proxy server*, l'adresse IP reçue par le serveur sera celle du *proxy server* et non celle du client. Cette méthode de contrôle ne contrôle donc que l'accés d'une machine, mais pas d'une personne.

Selon le type de serveur, ces contrôles peuvent s'appliquer à une arborescence, à un répertoire ou à un fichier. Ces différents types de contrôle peuvent également être combinés.

Certains serveurs fournissent la possibilité de définir les paramètres de ces contrôles soit dans un fichier unique (`access.conf` par exemple), soit dans un fichier par directory (`.htaccess` ou `www_acl` par exemple), soit les deux. La centralisation de ces informations en un unique fichier, bien protégé, est à préférer, car le fichier local peut être accédé par son URL, comme un simple document, et donc consulté.

Il ressort de cette analyse que les contrôles d'accés fournis en standard par le protocole HTTP et les différents types de serveurs Web existant ne sont pas assez surs pour permettre

l'utilisation du Web comme interface d'applications sensibles, telles que le commerce électronique.

Les applications sensibles, de type commerce électronique, nécessitent un mécanisme d'authentification sûr: les deux parties échangeant une transaction doivent être sûres de l'identité de leur interlocuteur. Les techniques exposées ci-dessus montrent leur insuffisance à fournir cette garantie. De plus, la **confidentialité** des échanges a pour limites celles bien connues du support utilisé par le web, l'Internet.

Les Solutions Proposées

Les diverses solutions proposées reposent sur la cryptographie, ou *chiffrement*: les informations échangées sont codées de façon à ce que seule la connaissance d'une clé permet leur décodage. Ces solutions ont donc pour but de renforcer la sécurité des mécanismes d'**authentification**, de **confidentialité** et d'**intégrité**.

La jeunesse de l'apparition de ces problèmes fait que pour le moment aucun standard n'a encore été défini par l'organisme admis comme faisant autorité en la matière, le **WWW Consortium**. Cependant, les compagnies les plus actives dans le domaine ont déja proposé et mis en service des solutions. Nous citerons ici une liste non exhaustive des méthodes existantes et plus ou moins disponibles aujourd'hui.

SSL: Secure Socket Layer

C'est la solution proposée par la société **Netscape Communications**. Cette société a développé le protocole **Secure Socket Layer**, ou **SSL**.Ce protocole se place entre le protocole de l'application (HTTP pour le Web) et TCP/IP. Il fournit le chiffrement (et donc la confidentalité) des données, l'authentification du serveur pour le client, l'intégrité des messages échangés, et optionnellement l'authentification du client pour le serveur. Ce protocole peut donc s'appliquer également aux autres protocoles d'applications tels que SMTP, Telnet, FTP, NNTP... Le portage d'une application existante consiste à remplacer les appels aux fonctions de la librairie socket BSD par des appels aux fonctions équivalentes de la librairie SSSL.

Toutes les versions actuelles de *Netscape Navigator*, sur toutes les plateformes, intègrent ce protocole. Par contre, seule la version commerciale du serveur de Netscape, *Netscape Commerce Server*, intègre SSL.

L'intégration de SSL a conduit à définir une nouvelle méthode d'accés dans les URL, baptisée **https**, pour se connecter à un serveur en utilisant le protocole SSL en dessous du protocole HHTP. Le port par défaut est le port 443.

Le protocole SSL peut être résumé comme suit:
- le client se connecte au serveur, qui l'identifie comme un client SSL
- le serveur envoie un certificat d'identité, qui inclue sa clé publique
- le client génère une clé de session aléatoirement, et l'envoie au serveur encryptée avec la clé publique du serveur.
- la session est ensuite encryptée selon l'algorithme RC4 de RSA (sur 40 bits), HTTP étant alors parlé au niveau de l'application comme d'habitude.

Netscape Navigator signale à l'utilisateur que les transactions sont sécurisées au moyen d'une ligne bleue et par une icône représentant un clé intègre (la clé est cassée si la transaction n'est pas sécurisée). Le choix *Document Inormation* du menu *File* donne les informations sur l'authentification du serveur, le chiffrement utilisé...

Ce protocole offre l'avantage de ne pas avoir à demander à l'utilisateur du client d'avoir et de taper une clé privée. Il a été principalement défini pour permettre de transférér facilement et surement un numéro de carte de crédit via le Web. Il n'y a cependant pas moyen d'avoir un reçu signé du vendeur, et l'utilisation des proxy servers doit être modifiée.

La prochaine version de Netscape Navigator (version 2.0), annoncée pour Décembre 1995, propose entre autres une messagerie sécurisée. Selon des informations officieuses aujourd'hui, et à prendre avec les réserves qu'il se doit, Netscape Communicationaurait obtenu l'autorisation d'utiliser en France le chiffrement avec une clé de 40 bits.

Il est clair que seul un client intégrant SSL (Netscape Navigator) peut dialoguer avec un serveur SSL (Netscape Commerce Server).

Le chiffrage de Netscape a récemment été cassé, aux USA et en France par un chercheur de l'INRIA. La réponse de Netscape Communications est que ce n'est ni SSL ni le chiffrement qui sont en cause, mais l'algorithme de génération aléatoire de la clé de session, et que c'est donc ce qui sera amélioré.

SHTTP: Secure HTTP

Le protocole **Secure HTTP**, ou **SHTTP**, a été proposé par la société **EIT** (Enterprise Integration Technolgies). Il est soutenu par le consortium CommerceNet (Silicon valley, Californie). Les transactions sont chiffrées et signées.

SHTTP se place au niveau de l'application. Le procole HTTP est enrichi: les différents paramètres (encryptage, identification...) sont décrits par des attributs de la requête HTTP. De même, il a imposé des modifications au niveau du langage HTML: de nouvaux attributs dans la définition des ancres ont été créées (par exemple `CRYPTOPTS =...`). Le niveau de sécurité est donc attribué ponctuellement par le développeur du document; le client peut par exemple demander au serveur que sa réponse soit chiffrée et signée.

L'intégration de SHTTP a conduit à définir une nouvelle méthode d'accés dans les URL, baptisée **shttp**.

Plusieurs méthodes de chiffrement sont supportées: chiffrement de RSA avec clé publique, RC2, MD4...

Le client *Secure Mosaic* a été développé par EIT en collaboration avec RSA et le NCSA. La société *Terisa Systems*, cofondée par EIT et RSA, fournit des boîtes à outil logicielles pour l'intégration de SHTTP. Les membres du consortium CommeceNet projetent d'utiliser ce système pour leurs transactions.

Secure Mosaic gère une base de données de clés publiques et certificats des serveurs intéressant l'utilisateur. Ces bases de données, locales à la machine client, sont chiffrées avec un mot de passe défini par l'utilisateur. Secure Mosaic peut également générer des paires de clés publiques et privées.

Contrairement à Netscape Navigator, l'interface utilisateur a du être enrichie pour offrir les zones de dialogue nécessaires.

OSF DCE

Open Software Foundation a proposé une sécurisation du Web basée sur sa technologie **Distributed Computing Environment**. Le système OSF/DCE est un vaste édifice offrant des

services tels que contrôle d'accés, authentification, chiffrement (méthode DES avec clé privée), intégrité (MD5: résumé de messages) et d'autres outils destinés à l'administration de systèmes, notamment à l'échelle de l'entreprise et du groupe de travail.

La boîte à outil DCE Web se place entre le protocole de l'application (HTTP) et le protocole de communication (TCP, UDP...).

Le but du projet DCE Web est de fournir ces services de sécurité de façon transparente à l'utilisateur familier du Web. Des passerelles ont été implémentées pour permettre la communication avec des serveurs n'étant pas dans un environnement DCE.

Le portage de NCSA Mosaic a nécessité la modification de 50 lignes de code sur le serveur NCSA HTTPD, et 70 sur le client NCSA Mosaic, plus l'ajout d'un peu d'interface utilisateur.

HTTP 1.5: Kerberos

Le NCSA travaillait en 1994 sur l'intégration du système **Kerberos** dans le protocole HTTP.

Kerberos, développé au MIT, est basé sur un service *tierce partie* fournissant des tickets d'accés aux clients. Les tickets ainsi obtenus par le client et envoyés au serveur contiennent une clé de session. Le chiffrement utilisé est symmétrique (DES). Un canal chiffré est ainsi établi entre le client et le serveur, qui s'authentifient ainsi mutuellement.

L'état d'avancement de ces travaux n'est pas connu.

La législation aux USA et en France

Les aspects légaux ne simplifient pas le problème.

Une loi fédérale américaine interdit à toute société américaine non seulement d'exporter hors USA, Canada et Mexique des logiciels de chiffrement dont la clé excède 40 bits, mais interdit également l'échange de données chiffrées avec une clé supérieure à 40 bits.

En France, l'échange de données chiffrées est considéré comme une arme de guerre, et est à ce titre soumis à autorisation. Lutilisation du chiffrement est autorisée uniquement pour la phase d'authentification aprés déclaration au Service Central de la Securité des Systèmes d'Information (SCSSI, Issy les Moulineaux, 1-40 95 37 15). Des demandes d'autorisation de chiffrement peuvent cependant y être déposées, en indiquant la méthode de chiffrement utilisée.

Les Solutions Utilisées et le futur

Beaucoup de solutions sont donc proposées, développées et testées dans le but de pouvoir héberger sur le Web de plus en plus d'applications dans des conditions opérationnelles et satisfaisantes. Des applications de plus en plus sensibles, comme le commerce électronique, la fourniture et la facturation de services, ou même la soumission de déclaration d'impots (projet en cours d'étude aux USA) sont en train de voir le jour, et seront peut être monnaie courante dans un futur plus ou moins proche. Les enjeux économiques sont donc importants, ce qui rend la situation assez confuse pour le moment.

La solution proposée aujourd'hui par Netscape Communication (SSL) a le mérite d'exister, d'être opérationnelle et simple pour l'utilisateur; elle répond à une partie du problème (échanges

de numéros de cartes de crédit). La gratuité et la disponibilté multi-plateformes du client sécurisé facilitent de plus sa pénétration du marché. De nombreux *Cyber Shops* l'ont d'ores et déja adoptée aux USA.

Nous n'avons pas d'informations récentes sur l'utilisation des autres solutions proposées. Aux USA, le système *e-cash*, développé par la société DigiCash, semble remporter un certain succés. Ce système de payement électronique n'utilise pas le protocole HTTP, et emploie des techniques de signature électronique à clé publique.

L'évolution de la sécurité sur le Web est également conditionnée par la façon dont évoluera la sécurité de son support de communication, l'Internet.

Le congrés international WWW qui doit se tenir en Décembre 1995 à Boston offrira peut être au Consortium W3 l'occasion de définir sa position sur le problème, et de jouer en quelque sorte le rôle d'arbitre qui lui revient.

Pour en savoir plus

- HTTP Security group of W3C
  `http://www.W3.org/pub/WWW/Security/Overview.html`
- The WWW Security FAQ
  `http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html`
- WWW Access Authorization protocol Examples
  `http://www.w3.org/hypertext/WWW/AccessAuthorization/ProtocolExamples.html`
- Netscape Security
  `http://home.netscape.com/info/security-doc.html`
- Netscape (In)Security
  `http://hplyot.obspm.fr/~dl/netscapesec/`
  `http://pauillac.inria.fr/~doligez/ssl/`
- Netscape SSLRef 2.0
  `http://home.netscape.com/newsref/std/sslref.html`
  `http://home.netscape.com/newsref/std/SSL.html`
- Netscape 2.0 Security
  `http://home.netscape.com/comprod/products/navigator/version_2.0/security.html`
- Un exemple de page securisée, à consulter avec Netscape
  `https://www6.internet.net/cgi-bin/getNode?node=20667#334617`
- EIT Secure HTTP
  `http://www.eit.com/creations/s-http/`
- Secure NCSA Mosaic Manual
  `http://www.commerce.net/software/SMosaic/Docs/manual.html`
- DCE Web Home Page

http://www.osf.org/www/dceweb/DCE-Web-Home-Page.html

- Web Servers Comparison
  http://www.proper.com/www/servers-chart.html
- Le Chiffrement en France
  http://web.cnam.fr/Network/Crypto/
- Netscape Security Issue
  http://home.netscape.com/newsref/std/random_seed_security.html
- Kerberizing the Web
  http://snapple.ncsa.uiuc.edu/adam/khttp/intro.htm
  http://snapple.ncsa.uiuc.edu/adam/kerbstuff.html
- Java: the Inside story
  http://www.Sun.COM/sunworldonline/swol-07-1995/swol-07-java.html
- CommerceNet.
  http://www.commerece.net
- DigiCash.
  http://www.digicash.com/
- Serveur ftp de l'Université de Eindhoven, Pays Bas.
  ftp://ftp.win.tue.nl/pub/security