

SSL/TLS : état des lieux et recommandations

Olivier Levillain

ANSSI

6 juin 2012



- 1 Introduction
- 2 Détails du protocole
 - a. Versions de SSL/TLS
 - b. Suites cryptographiques
 - c. Certificats
- 3 Écosystème HTTPS
 - a. Versions de SSL/TLS
 - b. Suites cryptographiques
 - c. Certificats
- 4 Conclusion

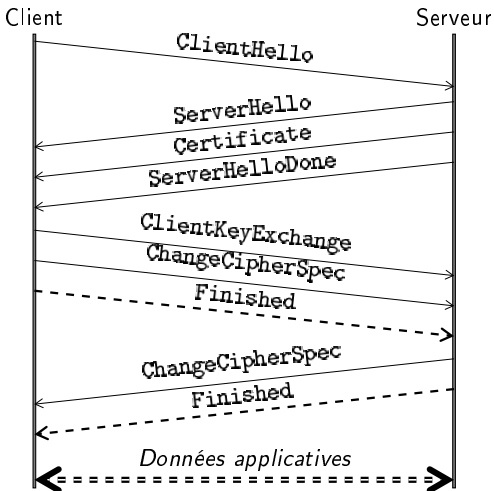


SSL/TLS : une brique essentielle d'Internet

- `https://` inventé par Netscape en 1995
 - début du commerce en ligne
- Utilisation massive aujourd'hui
 - HTTPS, bien au-delà du commerce en ligne
 - Sécurisation d'autres protocoles (SMTP, IMAP, LDAP, etc.)
 - VPN SSL
 - EAP TLS
- SSL (*Secure Sockets Layer*) ou TLS (*Transport Layer Security*) ?



Fonctionnement du protocole (SSLv3 et TLSv1.*)



1 Introduction

2 **Détails du protocole**

- a. Versions de SSL/TLS
- b. Suites cryptographiques
- c. Certificats

3 Écosystème HTTPS

- a. Versions de SSL/TLS
- b. Suites cryptographiques
- c. Certificats

4 Conclusion



SSLv2 : une version à proscrire

- SSLv2 publié en 1995
- Protocole considéré comme dangereux
- RFC 6176 en mars 2011 (*Prohibiting SSLv2*)
 - HMAC MD5 pour l'intégrité
 - partage de la clé pour l'intégrité et la confidentialité
 - mauvaise gestion de la fin de connexion
 - possibilité de faire une négociation à la baisse des algorithmes

Une bonne nouvelle : SSLv2 tend (enfin) à disparaître.



SSLv3 : une mise à jour majeure

- SSLv3 publié en 1996

- Deux gros soucis avec de vieilles implémentations SSLv3 :
 - attaque de Bleichenbacher en 1998 sur PKCS#1
 - incompatibilité avec les extensions TLS (RFC 3546, 2003)
 - ▶ courbes elliptiques
 - ▶ reprise de session sans état côté serveur
 - ▶ renégociation sécurisée

- Une implémentation corrigeant ces deux défauts implémentera en pratique TLSv1.0
 - SSLv3 ne présente donc aucun intérêt aujourd'hui



TLV1.0 : la normalisation par l'IETF

- En 2001, le protocole est repris par l'IETF et devient TLSv1.0 sans changement fondamental
- Attaque de Rogaway en 2002 sur le mode CBC avec IV implicite...
- médiatisée en 2011 par Duong et Rizzo (BEAST)
- Réactions diverses
 - RC4 en priorité
 - bricolage pour randomiser les IVs (en éclatant les messages)
 - passer à TLSv1.1



TLV1.1 et 1.2 : en route vers le futur

- TLV1.1 : version minimum conseillée

- TLV1.2 est une refonte du standard
 - inclusion des extensions dans le standard
 - inclusion d'autres RFCs dans le cœur du standard

 - avec quelques petits cadeaux pour les algorithmes cryptographiques
 - ▶ modes combinés pour le chiffrement et l'intégrité (GCM)
 - ▶ ajout de suites cryptographiques avec HMAC SHA256
 - ▶ possibilité d'utiliser SHA2 pour dériver les clés



Description d'une suite cryptographique

Une suite cryptographique décrit les algorithmes

- d'authentification (du serveur)
- d'échange de clé
- de chiffrement des données
- de protection en intégrité des données

Traditionnellement, on regroupe

- les parties K_x et A_u d'une part
- les parties Enc et Mac d'autre part



Exemples

TLS_RSA_WITH_RC4_128_MD5

- RSA : chiffrement RSA (échange de clé et authentification implicite)
- RC4_128 pour le chiffrement des données
- HMAC-MD5 pour l'intégrité des données

TLS_DHE_RSA_WITH_AES_128_CBC_SHA

- DHE : échange de clé Diffie-Hellman...
- RSA : signé par RSA
- AES_128_CBC pour le chiffrement des données
- HMAC-SHA1 pour l'intégrité des données

DHE/ECDHE assurent la PFS (*Perfect Forward Secrecy*)



Fonctionnement de la négociation

- Le client propose une liste de suites (ClientHello), selon ses préférences
- Le serveur choisit parmi cette liste
- Deux cas classiques :
 - comportement courtois (Apache)
 - comportement directif (IIS)



Message Certificate

- L'authentification serveur est toujours nécessaire en pratique
- Elle repose majoritairement sur des certificats X.509
- Le message Certificate contient une suite de certificats
 - le premier désigne le serveur TLS
 - chaque certificat est signé par le suivant
 - l'autorité racine peut être omise

- L'authentification client existe mais est négociée indépendamment des suites cryptographiques
 - des certificats X.509
 - deux nouveaux messages



1 Introduction

2 Détails du protocole

- a. Versions de SSL/TLS
- b. Suites cryptographiques
- c. Certificats

3 Écosystème HTTPS

- a. Versions de SSL/TLS
- b. Suites cryptographiques
- c. Certificats

4 Conclusion



Test des navigateurs

- Lancement de 5 serveurs SSL/TLS n'acceptant chacun qu'une seule version



Test des serveurs HTTPS

- Utilisation des données issues de l'EFF
- Énumération des serveurs HTTPS dans l'espace IPv4
 - 11 millions de serveurs HTTPS
 - dont 3,4 millions de serveurs reconnus valides par Firefox
- Face à un ClientHello SSLv2 de compatibilité :
 - 95 % TLSv1.0
 - 5 % SSLv3
 - quelques réponses SSLv2
- et pour les serveurs présentant un certificat de confiance :
 - 99 % TLSv1.0
 - 1 % SSLv3



Implémentations

Logiciel	SSLv2	SSLv3	TLSv1.0	TLSv1.1	TLSv1.2
OpenSSL				1.0.1	1.0.1
GnuTLS					
NSS					
IE sous 7					
IE sous XP					
Firefox					
Chrome				21	
Opera					
Safari					
IIS sous 2008					
IIS sous 2003					
Apache mod_ssl				1.0.1	1.0.1

Non supporté
Configurable
Supporté

- TLSv1.1 et TLSv1.2 pas toujours activés par défaut



Tests de capacités

- Côté navigateurs :
 - lancement de n serveurs SSL/TLS n'acceptant chacun qu'une seule suite
- Côté serveurs :
 - envoi de n ClientHello pour tester toutes les suites acceptées
 - sslscan permet de faire cela
 - (démarche intrusive)
- Presque tous supportent des suites conformes à des référentiels comme le RGS (algorithmes, tailles de clés, *PFS*)



Réalité des connexions

- Statistiques sur les données de l'EFF :
 - 38 % TLS_DHE_RSA_WITH_AES_256_CBC_SHA
 - 32 % TLS_RSA_WITH_RC4_128_MD5
 - 23 % TLS_RSA_WITH_AES_256_CBC_SHA
- Conséquence du comportement directif de certains serveurs
 - IIS choisira toujours TLS_RSA_WITH_RC4_128_MD5
 - le comportement est configurable, mais *global* au système sous Windows
- Est-il possible d'imposer la PFS du point de vue du client ?
 - ne proposer que des suites DHE/ECDHE fait perdre des serveurs
 - cette configuration est globale à l'application, voire au système
 - besoin d'affiner la décision par application et par serveur



Test des navigateurs

- Possibilité de monter un serveur TLS pour chaque test
 - bon usage des extensions X.509
 - vérification correcte du nom de domaine
 - vulnérabilité sur les noms contenant des caractères nuls
 - etc.
- Plate-forme partielle sur
<https://ssltest.offenseindepth.com/>



Les magasins de certificats

- Certaines applications utilisent un magasin indépendant
 - produits Mozilla
 - Opera
 - applications Adobe
 - Java (parfois)
- D'autres utilisent des magasins partagés
 - applications Microsoft
 - Safari
 - Chrome
 - produits Mozilla, à l'avenir (`$HOME/.pki`)
- Pour ou contre un magasin centralisé ?
 - avantages : retrait efficace d'une autorité compromise, ajout rapide d'autorités internes
 - inconvénient : manque de finesse dans la politique de confiance



Données de l'EFF

- Seuls 60 % des messages Certificate sont conformes
 - chaînes incomplètes
 - chaînes désordonnées
 - certificats dupliqués
 - certificats inutiles
 - chaînes multiples

- Incompatibilités réelles entre certaines piles et certains sites



Conclusion

- SSL/TLS est un protocole mûr basé sur des schémas éprouvés et il est théoriquement possible de l'employer correctement
- Si on maîtrise clients et serveurs, on peut le mettre en pratique de manière sécurisée
- Pour les navigateurs, beaucoup de difficultés
 - équilibre difficile entre compatibilité et sécurité
 - mécanisme de gestion de la confiance en tout ou rien
- Perspectives :
 - proposer des plates-formes de test (client/serveur)
 - continuer l'analyse du paysage SSL
 - travail sur la confiance et la révocation (CA/B Forum, OCSP stapling, DANE, TACK...)
 - IDS/IPS : utilisation de Suricata
 - proxy local pour affiner la gestion de la confiance



Questions ?

Merci de votre attention.

