SSL/TLS

SECURE SOCKETS LAYER - TRANSPORT LAYER SECURITY

Anaïs Gantet, Benoît Camredon

Plan

Introduction

Protocoles non sécurisés

Beaucoup de protocoles non sécurisés

- `HTTP`
- `SMTP`, `IMAP`
- `FTP`
- ...

Nouveaux risques, nouveaux besoins

- De plus en plus d'attaques
- Des besoins de plus en plus risqués : commerce électronique...

Un client et un serveur peuvent avoir une communication sécurisée sur un réseau non sécurisé, alors qu'ils ne s'étaient jamais rencontrés auparavant.

SSL/TLS : Mécanismes de sécurité

Objectif : Sécuriser les communications

- Confidentialité
  - Eviter l'écoute passive/active
- Authentification
  - Etre sûr de l'entité avec qui on communique
- Intégrité
  - Etre capable de détecter si un message a été modifié

Comment ?

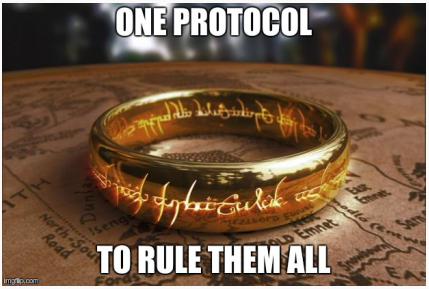
- Confidentialité
  - Chiffrement des communications (chiffrement symétrique)
- Authentification
  - Utilisation de certificats...
- Intégrité
  - Code d'authentification de message

Sécurisation de protocoles non sécurisés...

- Initialement pour `HTTP` → `HTTPS`
- Puis élargi à d'autres
  - `SMTP` → `SMTPS`
  - `IMAP` → `IMAPS`
  - `FTP` → `FTPS`
  - ...

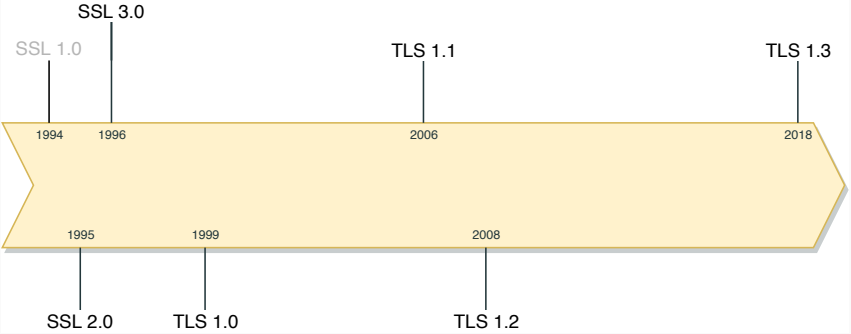
Mais aussi...

- Création de réseaux privés virtuels : `OpenVPN`
- Sécurisation de réseaux : `EAP-TLS`



Deux variantes d'un même protocole

- SSL (Secure Sockets Layer) : Netscape
- TLS (Transport Layer Security) : IETF



Fonctionnement du protocole

Etapes SSL/TLS

Différentes étapes

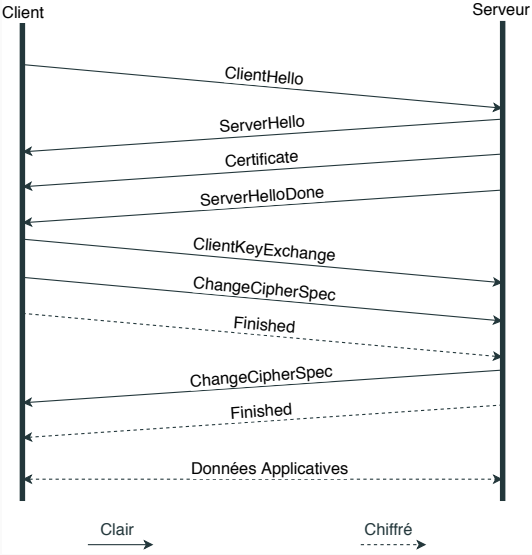
1. Négocier la suite cryptographique qui va être utilisée
2. Etablir les éléments secrets de session
3. Authentifier les parties
4. Transmettre les données applicatives de façon sécurisée (confidentialité/intégrité)

Plusieurs protocoles

- `Handshake Protocol`
  - Négociation de la suite cryptographique
  - S'accorder sur une clé maître
  - Authentifier les acteurs
- `Record Protocol`
  - Transmission des données applicatives
  - Chiffrement symétrique
- `Alert Protocol`
  - Quand il y a des problèmes

Messages SSL 3.0 / TLS1.[0,1,2]

- `ClientHello` : version SSL/TLS, Ensemble de suites cryptographiques, une suite d'octets aléatoires...
- `ServerHello` : version SSL/TLS choisie, suite cryptographique choisie, une suite d'octets aléatoires...
- `Certificate` : Certificat du serveur
- `ClientKeyExchange` : Génération d'une *pre master key* chiffrée avec la clé publique du serveur
- `ChangeCipherSpec` : Activation des paramètres négociés
- `Finished` : Termine la communication et contient un hashé de l'ensemble des messages échangés



PFS : Perfect Forward Secrecy

Définition

- Propriété cryptographique qui garantit que la découverte de la clé privée d'un des participants ne compromettra pas la confidentialité des communications passées
- Une nouvelle clé unique est générée pour chaque session

Algorithmes d'échange de clé qui ne fournissent pas la PFS

- `RSA`
- `ECDH`
- ...

Algorithmes d'échange de clé qui fournissent la PFS

- Ephemeral Diffie-Hellman (`DHE`)
- Elliptic curve Ephemeral Diffie–Hellman (`ECDHE`)

Description d'une suite cryptographique

Une suite cryptographique décrit les algorithmes de

- Echange de clé K_x
- Authentification A_u
- Chiffrement de données *Enc*
- Protection en intégrité des données *Mac*
- Dérivation pour TLS 1.2 et supérieur

Exemples

- `TLS_RSA_WITH_RC4_128_MD5`
  - `RSA` : Chiffrement RSA (échange de clé et authentication implicite)
  - `RC4_128` : Chiffrement des données
  - `HMAC-MD5` : Intégrité des données
- `TLS_DHE_RSA_WITH_AES_128_CBC_SHA`
  - `DHE` : Echange de clé Diffie-Hellman
  - `RSA` : Signé par RSA (authentification du serveur)
  - `AES_128_CBC` : Chiffrement des données
  - `HMAC-SHA1` : Intégrité des données

Description d'une suite cryptographique TLS 1.3

Suite cryptographique

- Algorithme de chiffrement
- Fonction de hachage pour dérivation des clés

Extensions

- Echange de clé (DHE, ECDHE)
- Mécanisme d'authentification (PSK, password, RSA, ECDSA, EdDSA...)

Exemple de suites cryptographiques

Avant TLS 1.3

TLS_NULL_WITH_NULL_NULL
TLS_RSA_WITH_NULL_SHA
TLS_KRB5_WITH_DES_CBC_SHA
TLS_PSK_WITH_NULL_SHA
TLS_DH_Anon_WITH_RC4_128_MD5
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

TLS 1.3

TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256
TLS_AES_128_CCM_SHA256

STARTTLS

Objectifs

- Permettre à une communication non sécurisée de monter un tunnel sécurisé (SSL ou TLS) pour continuer à s'échanger des données
- Ajout d'une *commande* STARTTLS
- Pas besoin de changer le port !

Etapes

1. Initiation de la communication sur le port habituel
2. Le serveur dit qu'il sait gérer le STARTTLS
3. Le client demande l'établissement d'un tunnel sécurisé avec la commande STARTTLS
4. Négociation SSL/TLS entre le client et le serveur
5. Echange des données applicatives dans la session SSL/TLS

Exemple de protocoles : IMAP:143 (IMAPS:993), SMTP:25 (SMTPS:465), LDAP:389 (LDAPS:636)...

Versions

SSL 2.0

- Failles conceptuelles importantes
  - Possibilité de faire une négociation à la baisse des algorithmes (downgrade attack)
  - HMAC-MD5 pour l'intégrité
  - Partage de clé pour l'intégrité et la confidentialité
  - Mauvaise gestion de fin de connexion
- RFC 6176 en mars 2011 : Prohibiting Secure Sockets Layer (SSL) Version 2.0
- DROWN en 2016

SSL 3.0

- Deux gros soucis avec de vieilles implémentations SSLv3
  - Attaque de Bleichenbacher en 1998 sur PKCS#1 v1.5
  - Incompatibilité avec les extensions de TLS (RFC 3546, 2003)
    - Courbes elliptiques
    - Reprise de session sans état côté serveur
    - renégociation sécurisée
- TLSv1.0 corrige ces deux défauts, donc aucun intérêt pour SSLv3
- POODLE en 2014

TLS 1.0

- Repris par l'IETF en 2001 sans changement fondamental
  - Changement du type de padding
  - Ne dissocie plus les erreurs de padding des erreurs de vérification
  - Ajout des extensions

TLS 1.1

- Révision publiée en 2006 pour répondre à certaines attaques sur les modes de chiffrement `CBC`

TLS 1.2

- Inclusion des extensions dans le standard
- Nouveaux algorithmes cryptographiques
  - `AEAD` : Mode combiné pour le chiffrement et l'intégrité appelé aussi *chiffrement authentifié* (`AES-GCM`)
  - Ajout de suites cryptographiques avec `HMAC-SHA256`
  - Possibilité d'utiliser `SHA2` pour dériver les clés

TLS 1.3

Grosse refonte de TLS

- Nettoyage de tous les éléments non sécurisés/utilisés
- Amélioration de la sécurité
- Chiffrement plus important du protocole
- Amélioration de la performance (0-RTT...)

Beaucoup d'algorithmes supprimés

- Echange de clé : RSA
- Chiffrement symétrique: RC4, 3DES, Camellia
- Modes de chiffrement : AES-CBC

Elements du protocole supprimés

- Compression TLS
- Renégociation TLS

Algorithmes d'échange de clé

- DHE & ECDHE
- PSK
- PSK avec (EC)DHE

Algorithmes de signature

- RSA
- ECDSA / EdDSA

TLS 1.3 : Beaucoup d'attaques à présent inefficaces

RC4

- Roos's Bias 1995
- Fluhrer, Martin & Shamir 2001
- Klein 2005
- Combinatorial Problem 2001
- Royal Holloway 2013
- Bar-mitzvah 2015
- NOMORE 2015

RSA-PKCS#1 v1.5

- Bleichenbacher 1998
- Jager 2015
- DROWN 2016

Renégotiation

- Marsh Ray Attack 2009
- Renegotiation DoS 2011
- Triple Handshake 2014

3DES

- Sweet32

Mode CBC

- Vaudenay 2002
- Boneh/Brumley 2003
- BEAST 2011
- Lucky13 2013
- POODLE 2014
- Lucky Microseconds 2015

Compression

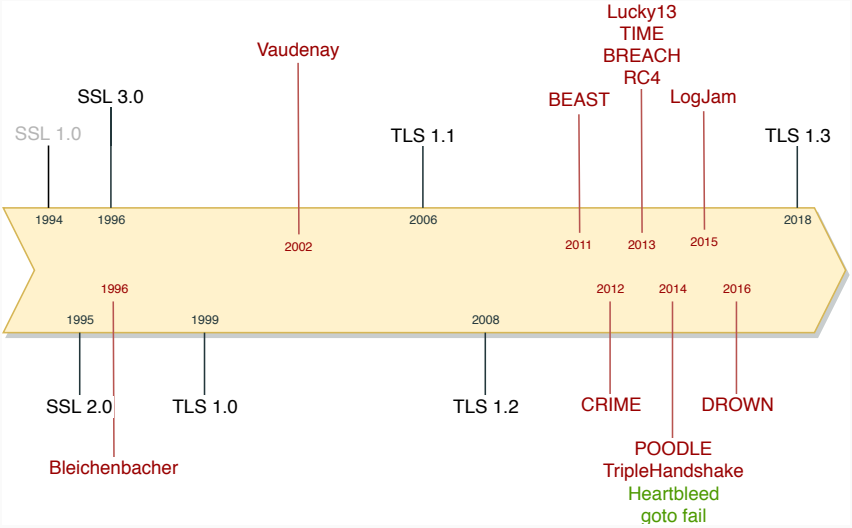
- CRIME 2012

MD5/SHA1

- SLOTH 2016
- SHAttered 2017

Attaques contre SSL/TLS

Timeline


Lucky13
TIME
BREACH
RC4
Vaudenay
SSL 3.0
BEAST
LogJam
SSL 1.0
TLS 1.1
TLS 1.3
1994
1996
2002
2006
2011
2013
2015
2018
1996
2012
2014
2016
1995
1999
2008
SSL 2.0
TLS 1.0
TLS 1.2
CRIME
DROWN
Bleichenbacher
POODLE
TripleHandshake
Heartbleed
goto fail

Pourquoi autant d'attaques ?

Utilisation massive sur Internet

- Plus de la moitié du trafic Internet est maintenant chiffré
- Besoin de confidentialité : censure, intimité...

=> **Intérêt pour les attaquants**

Objectifs complexes, contraintes fortes

- Réseau non sécurisé
- Communication antérieure ou secret partagé non nécessaire
- Protocoles/algorithmes différents entre client/serveur
- Puissance de calcul

=> **Compromis à faire**

Protocole complexe

- Nombreuses étapes : Machine à état complexe
- Nombreux algorithmes
- Nombreuses extensions
- ...

=> **Surface d'attaque importante**

Intérêts divergeants

- Besoin de sécurité
- Besoin de performance
- Besoin de contrôle

=> **Encore des compromis**

Généralités

- Attaques visent dans la majorité des cas `HTTPS`
  - Objectif : Voler un cookie de session
- Attaques souvent théoriques
  - Contraintes plus ou moins fortes
    - Pouvoir envoyer du flux en clair dans la session
    - Exécuter un code js sur la machine
  - Ne fonctionnent pas sur toutes les implémentations
- Montrent clairement des faiblesses

Attaques contre SSL/TLS

Attaque sur PKCS#1 v1.5

- Million Message Attack de Bleinchenbacher (1998)
  - Récupération de la clé de session en décryptant la communication RSA
    - Accès à la communication
    - Possibilité d'interagir avec le serveur
    - Faire la distinction entre une erreur de padding et une erreur de vérification
- DROWN : (Drown stands for Decrypting RSA with Obsolete and Weakened eNcryption)
  - Si le serveur supporte SSLv2
  - Faille openssl supplémentaire réduisant la complexité
  - Utilisation de SSLv2 pour décrypter des trames de TLS

Chiffrement RC4

- Très utilisé (WEP, TLS)
- Biais statistiques
  - Permet de retrouver un texte en clair s'il a été chiffré avec beaucoup de clés différentes
- Attaques: Fluhrer, Mantin and Shamir (2001), Klein (2005), Royal Holloway (2013), Bar-mitzvah (2015), NOMORE (2015)...
- Interdiction d'utiliser RC4 dans TLS suite à la RFC 7465 (2015)

Attaques contre SSL/TLS

Chiffrement par bloc CBC : Insertion de données

- BEAST (Browser Exploit Against SSL/TLS)
  - Vecteur d'initialisation
  - Concerne TLS 1.0, SSL 3.0... pour les chiffrements par bloc
  - Permet de savoir si un bloc de N octets est présent dans la communication
  - Nécessite de pouvoir envoyer du texte en clair dans la session SSL/TLS

Chiffrement par bloc CBC : Padding Oracle Attack

- Lucky 13 : TLS 1.0, SSL 3.0...
  - Distinguer une erreur MAC d'une erreur de padding (plusieurs tests à faire)
- POODLE : Padding Oracle On Downgraded Legacy Encryption
  - Attaque de l'homme du milieu
  - Possibilité de downgrader le protocole
  - Attaque sur SSL 3.0 : MAC-then-encrypt, Padding aléatoire
  - TLS_FALLBACK_SCSV

Compression

- CRIME, TIME, BREACH

goto fail : iOS

```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                 uint8_t *signature, UInt16 signatureLen)
{
  OSStatus        err;
  ...

  if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
  if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
  if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
  ...

fail:
  SSLFreeBuffer(&signedHashes);
  SSLFreeBuffer(&hashCtx);
  return err;
}
```

Heartbleed

Faille d'implémentation

- `openssl` de `1.0.1` à `1.0.1f` sont vulnérables
- Fuite du contenu de la mémoire permettant à l'attaquant de récupérer des informations de la machine
- Localisée dans l'extension `heartbeat` (Sorte de ping)

Etapes normales

1. Client : Envoie moi les 4 lettres `toto`
2. Serveur : `toto`

Attaque

1. Client : Envoie moi les 1000 lettres `toto`
2. Serveur : `toto\x00\x64secret\xfe...`

Interception SSL/TLS

Interception SSL/TLS

Objectif : Lire le contenu des communications `SSL/TLS` pour

- Censure
  - Interdiction d'aller voir certaines informations
  - Limiter les sites que l'on peut voir
- Protection des utilisateurs
  - Interdire les sites dangereux
  - Interdire les contenus dangereux
  - Maîtriser le trafic
- Protection de l'entreprise

Moyens

- Exploiter des vulnérabilités
- Négocier des algorithmes faibles
- Installer une CA sur les clients

Certificat

- Structure de données dans un format spécifique (X509)
- Certifie le lien clé publique / entité, par une autorité ou CA
- Contient des informations importantes
  - L'entité à qui appartient la clé publique
  - La CA qui certifie que la clé publique appartient à l'entité
  - Une date de validité
  - Des usages pour le certificat
  - ...

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 6448089797071798919 (0x597c3781ca490287)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = Google Trust Services,
                CN = Google Internet Authority G3
        Validity
            Not Before: Oct 30 13:14:00 2018 GMT
            Not After : Jan 22 13:14:00 2019 GMT
        Subject: C = US, ST = California, L = Mountain View,
                 O = Google LLC, CN = www.google.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:cb:45:6c:75:99:a4:5a:6d:94:a8:db:d9:cd:d1:
                    3c:a3:80:9e:82:d8:eb:36:0c:72:19:5d:2a:f9:19:
                    b5:a4:33:2c:e0:1d:1d:d3:51:b8:fa:ec:fb:bf:dd:
                    15:d2:92:3c:a0:51:71:57:66:58:a8:bc:9b:55:4f:
                    a0:3f:f8:74:00
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Key Usage: critical
                Digital Signature
            X509v3 Subject Alternative Name:
                DNS:www.google.com
            Authority Information Access:
                CA Issuers - URI:http://pki.goog/gsr2/GTSGIAG3.crt
                OCSP - URI:http://ocsp.pki.goog/GTSGIAG3
            X509v3 Subject Key Identifier:
                95:78:50:8C:BF:4F:14:85:F8:94:B9:D7:3C:B8:AB:EB:09:C2:D3:AB
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Authority Key Identifier:
                keyid:77:C2:B8:50:9A:67:76:76:B1:2D:C2:86:D0:83:A0:7E:A6:7E:BA:4B
            X509v3 Certificate Policies:
                Policy: 1.3.6.1.4.1.11129.2.5.3
                Policy: 2.23.140.1.2.2
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:http://crl.pki.goog/GTSGIAG3.crl
    Signature Algorithm: sha256WithRSAEncryption
         08:4a:a3:b8:87:dc:6e:f0:81:61:05:f0:3e:bc:75:05:14:7a:
         ...
```

Vérification du certificat

1. Récupérer le certificat sur le serveur (`Certificate`)
2. Vérifier que l'entitié du certificat correspond bien au serveur sur lequel on souhaite se connecter
3. Vérifier que le certificat est correct
   - Pas expiré
   - Usage cohérent
   - ...
4. Vérifier que le certificat est signé par une CA connnue
5. Vérifier que la signature est correcte

=> **Confiance aveugle dans l'autorité (pré-enregistrée dans les clients)**

Authentification du serveur

1. Récupération du challenge venant du serveur
2. Déchiffrement du challenge avec la clé publique. Si ça fonctionne, le serveur a bien la clé privée correspondant à la clé publique du certificat

Autorité de certification

Entités reconnues

- Une grosse partie d'Internet repose sur ces entités
- Certificat enregistré dans nos clients web (`/etc/ssl/certs`)
- Exemple : Verisign, Thawte...

Si une autorité se fait pirater ?

- L'attaquant pourra signer des certificats qui seront acceptés par les navigateurs
- Cas de *DigiNotar* en 2011

Public Key Pinning

- Clés publiques (serveur/autorité) en mémoire dans le client
- Limite le risque en cas de compromission d'une CA
- Mais possibilité d'utiliser une CA locale...

Bonus

Vos commentaires

Votre avis compte

- remarques ?
- améliorations ?
- suggestions ?
- questions ?

anais.gantet.pro@gmail.com
benoit.camredon@gmail.com

Pour aller plus loin...

Exemple de connection TLS illustrée et commentée

- https://tls.ulfheim.net

SSL/TLS

- https://www.sstic.org/media/SSTIC2012/SSTIC-actes/ssl_tls_soa_recos/SSTIC2012-Article-ssl_tls_soa_recos-levillain_2.pdf
- https://www.sstic.org/media/SSTIC2015/SSTIC-actes/ssltls_soa_reloaded/SSTIC2015-Article-ssltls_soa_reloaded-levillain_cObDbqp.pdf

TLS 1.3

- https://www.owasp.org/images/9/91/OWASPLondon20180125_TLSv1.3_Andy_Brodie.pdf

Attaques

- BEAST: https://commandlinefanatic.com/cgi-bin/showarticle.cgi?article=art027
- Lucky 13: http://www.isg.rhul.ac.uk/tls/Lucky13.html
- POODLE: https://www.dfranke.us/posts/2014-10-14-how-poodle-happened.html