Regards critiques sur SSL/TLS

Olivier Levillain

ANSSI

Séminaire sur la Sécurité Numérique

Qui suis-je ?

Olivier Levillain

- stage de DEA en cryptographie sur une fonction de hachage
- membre du laboratoire « système » de l'ANSSI (2007-2012)
- responsable du laboratoire « réseau » de l'ANSSI (2012-2015)
- responsable du CFSSI, centre de formation de l'ANSSI (2015-)

Recherche

- participation aux travaux sur les mécanismes bas-niveau x86
- travaux sur SSL/TLS (doctorant depuis 2011)
- études sur les langages depuis 2007
- travaux sur les *parsers*

Enseignement

- cours crypto sur les fonctions de hachage et la cryptanalyse
- cours système au CFSSI
- interventions sur SSL/TLS, et plus récemment sur la sécurité du développement

ANSSI

L'ANSSI (agence nationale de la sécurité des systèmes d'information) a des missions de protection des systèmes d'information :

- détecter et réagir aux attaques informatiques
- soutenir le développement de produits et services de sécurité de confiance
- fournir un conseil et un soutien dans le domaine
- communiquer sur les menaces pesant sur les systèmes d'information et sur les moyens de protection associés

Les publics visés sont :

- les ministères
- les entreprises
- le grand public

TLS en quelques mots

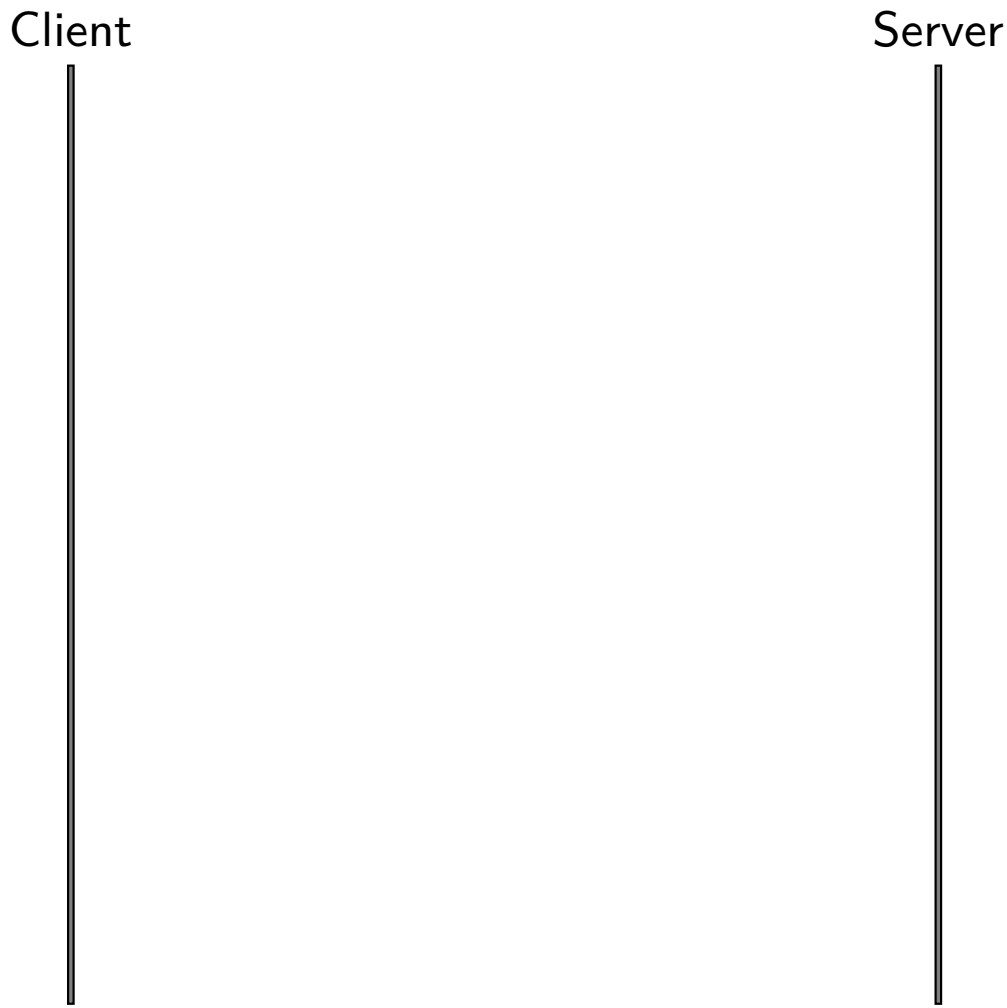
SSL/TLS : un pilier de la sécurité d'Internet

- Le schéma `https://` inventé par Netscape en 1995
  - début du commerce en ligne
- Omniprésence de SSL/TLS aujourd'hui
  - HTTPS, bien au-delà du commerce en ligne
  - Une méthode générique pour sécuriser d'autres protocoles (SMTP, IMAP, LDAP...)
  - VPN SSL
  - EAP TLS

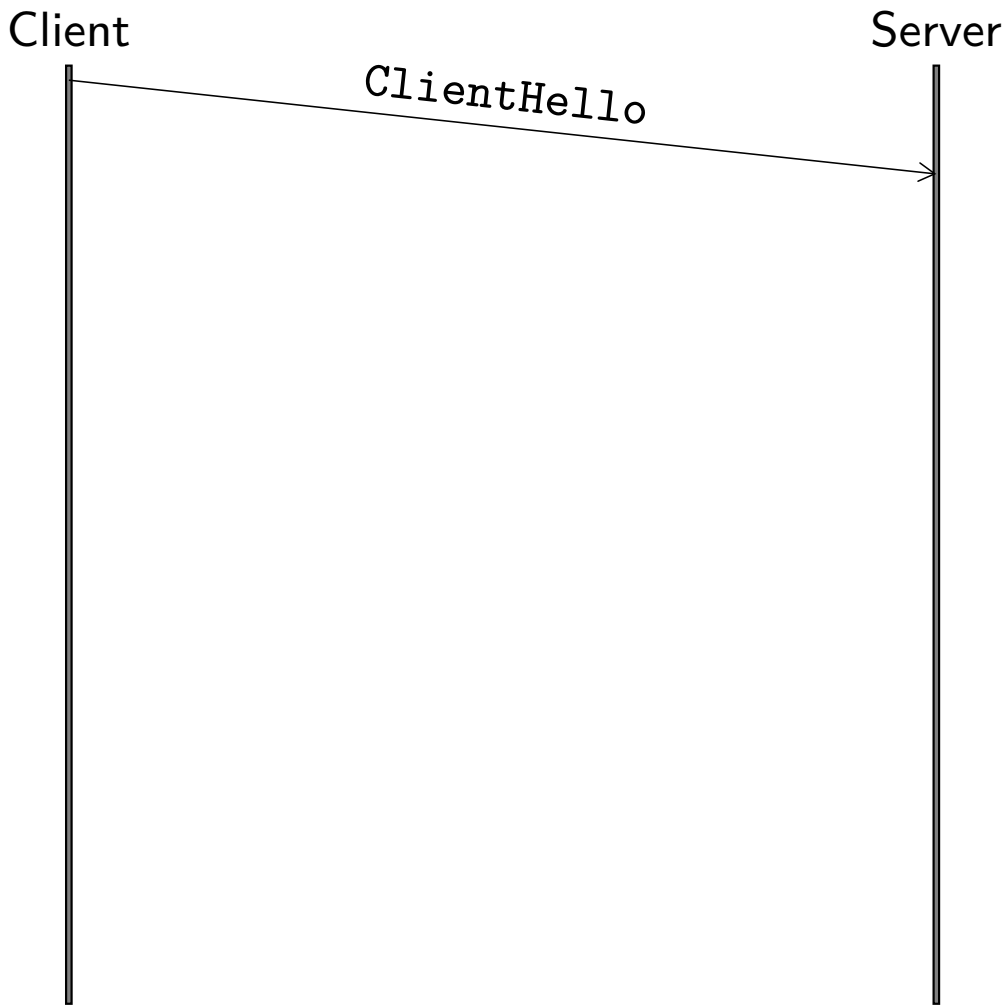
SSL/TLS : un pilier de la sécurité d'Internet

- Le schéma `https://` inventé par Netscape en 1995
  - début du commerce en ligne
- Omniprésence de SSL/TLS aujourd'hui
  - HTTPS, bien au-delà du commerce en ligne
  - Une méthode générique pour sécuriser d'autres protocoles (SMTP, IMAP, LDAP...)
  - VPN SSL
  - EAP TLS
- SSL (*Secure Sockets Layer*) ou TLS (*Transport Layer Security*) ?
  - SSLv2 (1995) et v3 (1996) conçu par Netscape
  - TLS 1.0 (2001) ou SSLv3.1, maintenu par l'IETF
  - De nouvelles révisions depuis : 1.1 (2006), 1.2 (2008) et 1.3 (2016 ?)

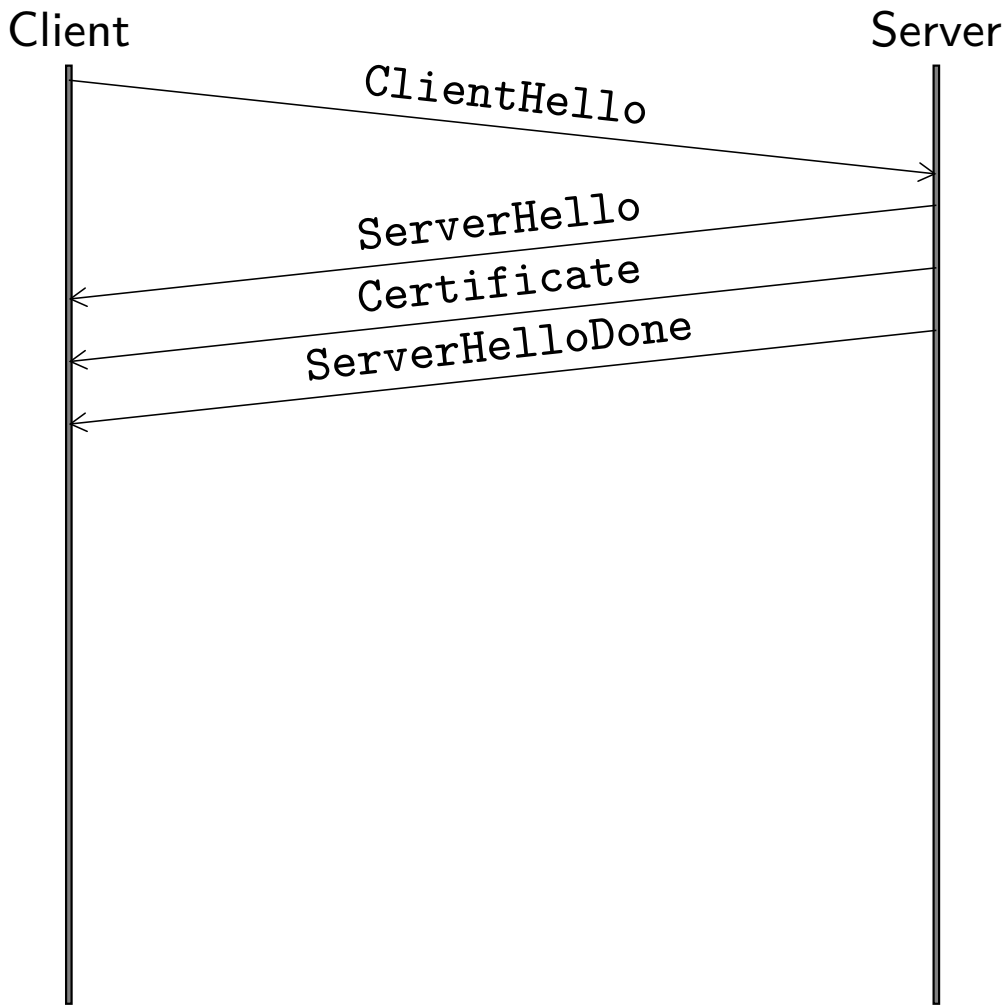
Fonctionnement du protocole



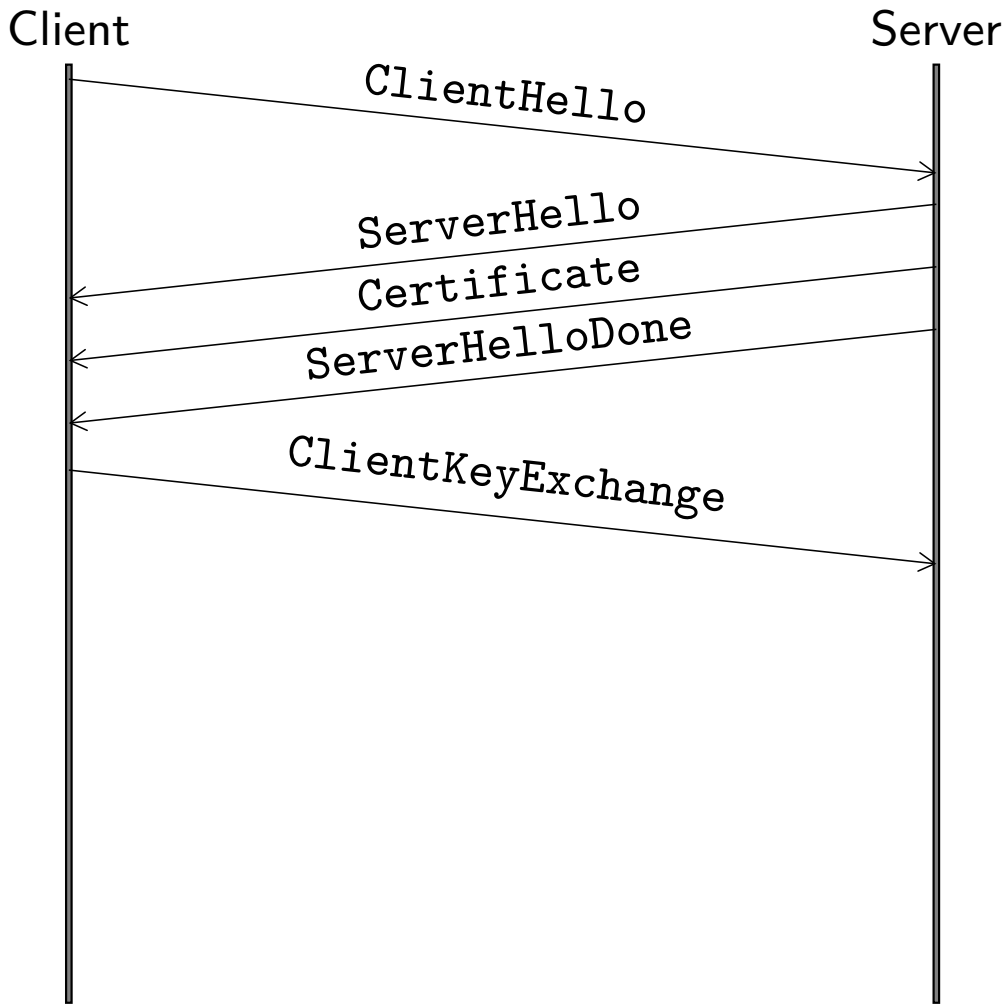
Fonctionnement du protocole



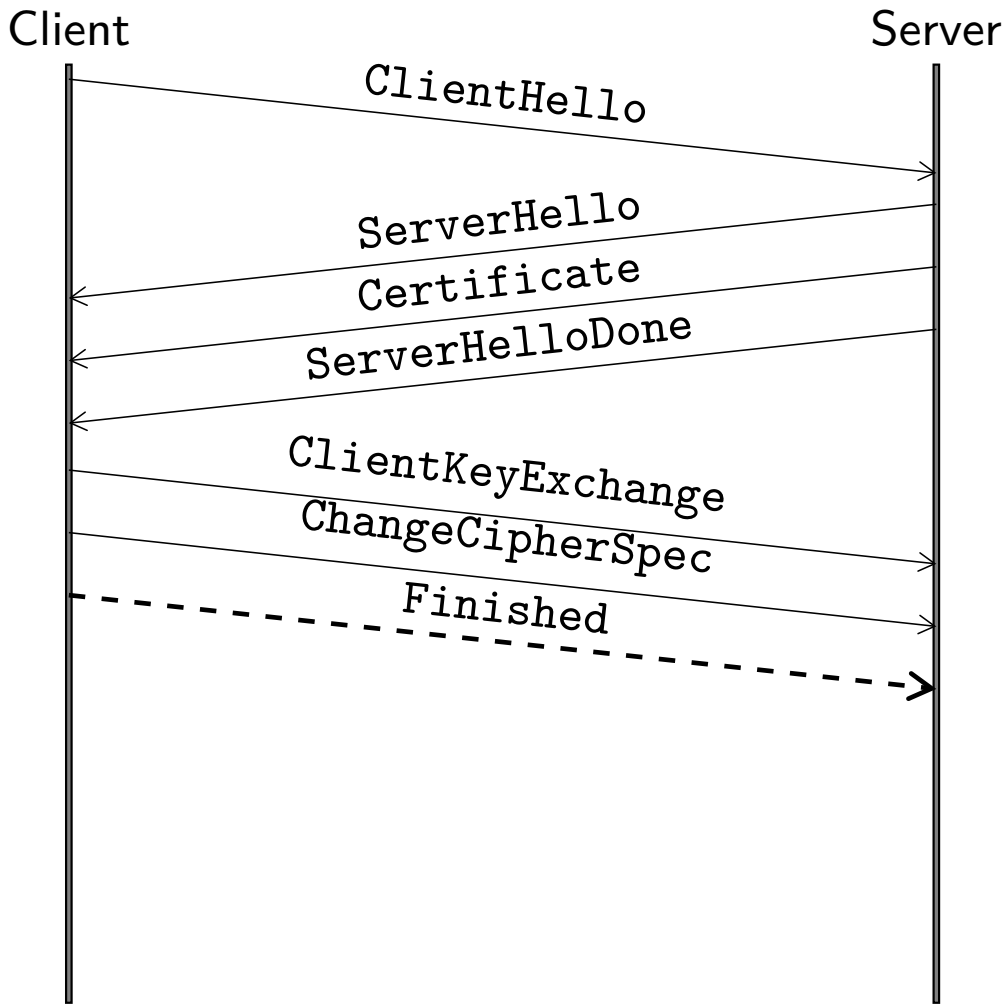
Fonctionnement du protocole



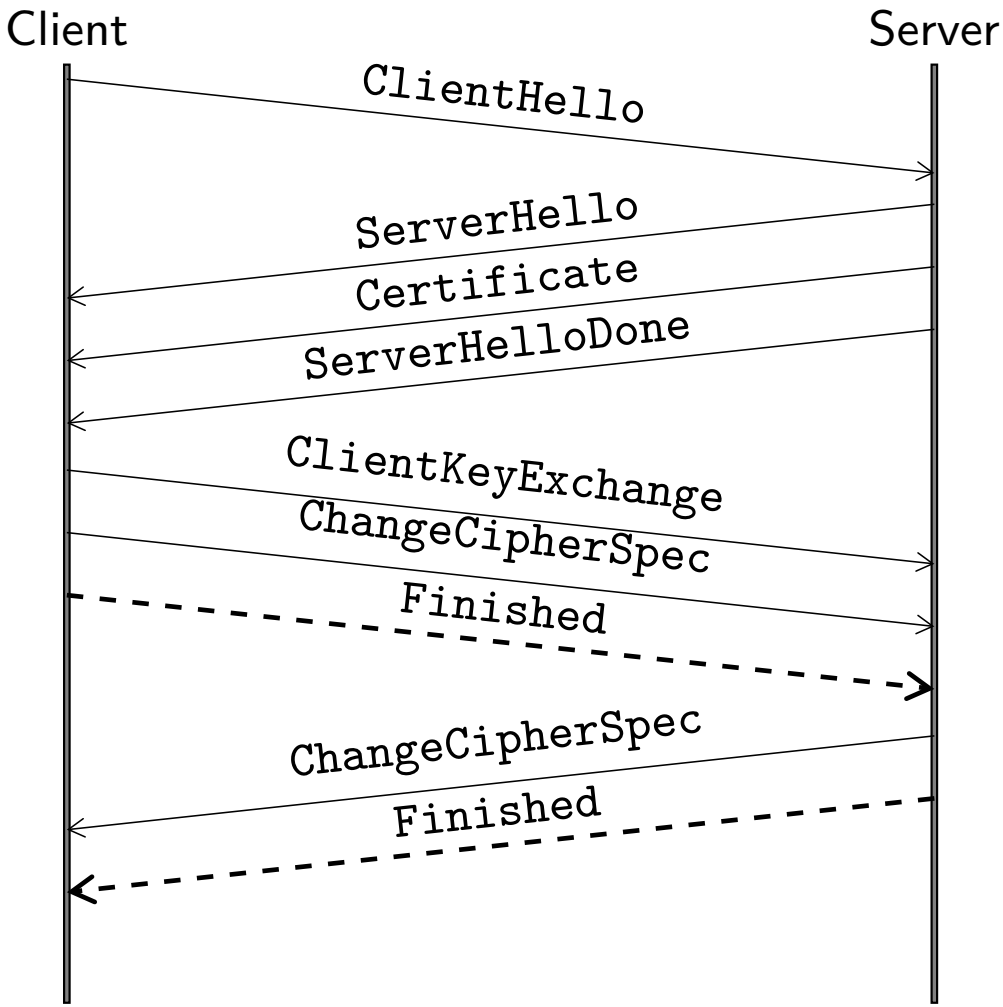
Fonctionnement du protocole



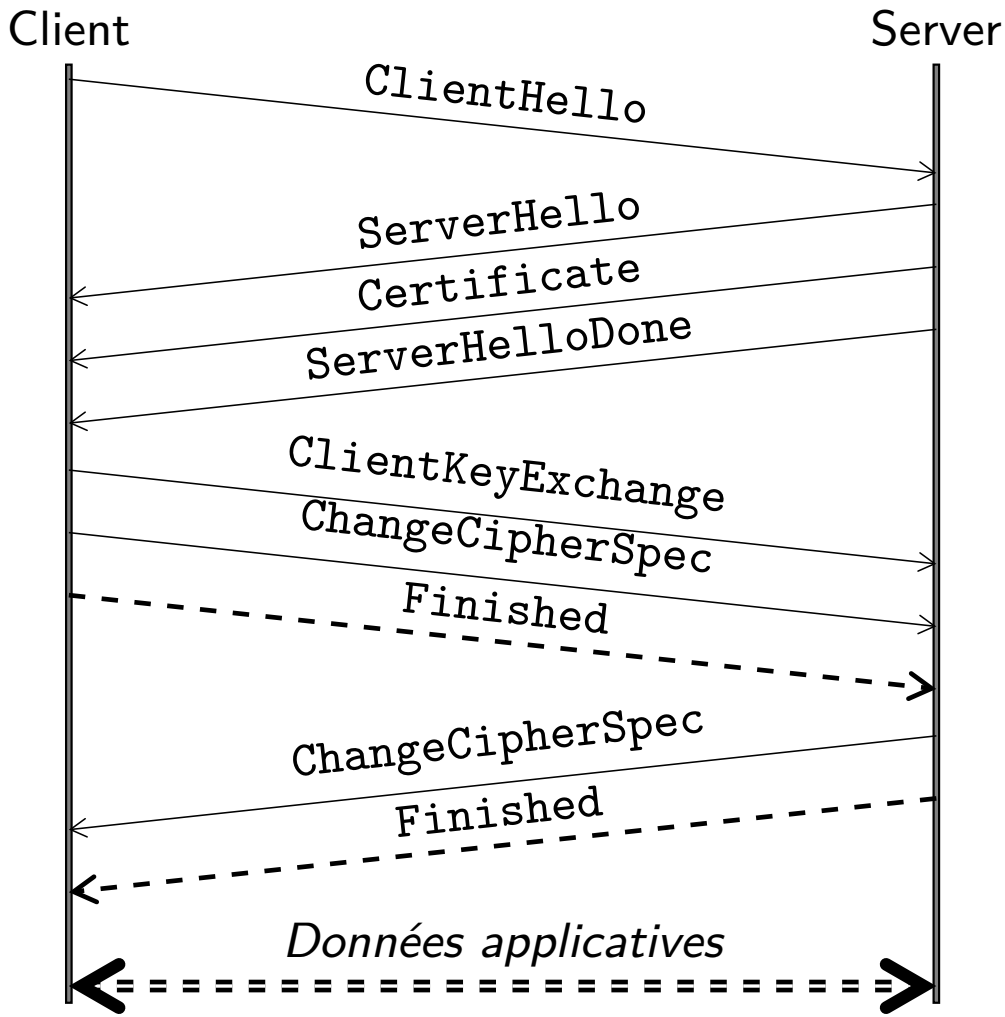
Fonctionnement du protocole



Fonctionnement du protocole



Fonctionnement du protocole



Quelques chiffres sur SSL/TLS

- Plus de 50 RFC
- 5 versions du protocole pour le moment
- Plus de 300 suites cryptographiques
- Plus de 20 extensions
- Quelques fonctionnalités *intéressantes*
  - compression
  - renégociation
  - reprise de session (2 méthodes)
- Une douzaine d'implémentations bien connues
- Et combien de piles maison ?

Les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : `AES128-SHA` et `ECDH-ECDSA-AES128-SHA` ?

Les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : `AES128-SHA` et `ECDH-ECDSA-AES128-SHA` ?

A `AES128-SHA`

Les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : `AES128-SHA` et `ECDH-ECDSA-AES128-SHA` ?

A `AES128-SHA`

B `ECDH-ECDSA-AES128-SHA`

Les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : `AES128-SHA` et `ECDH-ECDSA-AES128-SHA` ?

A `AES128-SHA`

B `ECDH-ECDSA-AES128-SHA`

C une alerte

Les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : `AES128-SHA` et `ECDH-ECDSA-AES128-SHA` ?

A `AES128-SHA`

B `ECDH-ECDSA-AES128-SHA`

C une alerte

D la réponse D (`RC4_MD5`)

Les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : `AES128-SHA` et `ECDH-ECDSA-AES128-SHA` ?

A `AES128-SHA`

B `ECDH-ECDSA-AES128-SHA`

C une alerte

D la réponse D (`RC4_MD5`)

Le pire, c'est qu'on peut l'expliquer

- une suite cryptographique est représentée par un entier sur 16 bits
- pendant longtemps, toutes les suites étaient de la forme `00 XX`
- pourquoi perdre du temps à regarder l'octet de poids fort ?

Les piles SSL/TLS maison (2/3)

- En 2010, Google propose plusieurs extensions, *False Start* et *Snap Start*
- Après plusieurs mois, Internet se montre intolérant à *Snap Start*
- L'expérimentation est abandonnée en 2012

Les piles SSL/TLS maison (2/3)

- En 2010, Google propose plusieurs extensions, *False Start* et *Snap Start*
- Après plusieurs mois, Internet se montre intolérant à *Snap Start*
- L'expérimentation est abandonnée en 2012

- Un an plus tard, le même problème resurgit dans un autre contexte
- Sur la liste de diffusion IETF (`tls@ietf.org`), on apprend finalement la raison : le `ClientHello` est trop gros...

Les piles SSL/TLS maison (3/3)

Examinons le début d'un `ClientHello` de 258 octets

16	03	01	01	02

TLS

Type	Version	Longueur
HS	*TLS 1.0*	*258*

SSLv2

Longueur	*Pad.*	Type
5635	...	*CH*

Un `ClientHello` TLS dont la taille est comprise entre 256 et 511 peut être confondu avec un `ClientHello` SSLv2 !

Les piles SSL/TLS maison (3/3)

Examinons le début d'un `ClientHello` de 258 octets

16	03	01	01	02

TLS

Type	Version	Longueur
HS	*TLS 1.0*	*258*

SSLv2

Longueur	*Pad.*	Type
5635	...	*CH*

Un `ClientHello` TLS dont la taille est comprise entre 256 et 511 peut être confondu avec un `ClientHello` SSLv2 !

Tout est bien qui finit bien

- Google a finalement proposé une extension pour ajouter du bourrage au `ClientHello`...

Un rapide aperçu de quelques vulnérabilités SSL/TLS

- 1995 : négociation à la baisse dans SSLv2
- 1998 : attaque de Bleichenbacher sur PKCS#1 v1.5
- 2002 : Mauvaise interprétation de l'extension X.509 *Basic Constraints* (IE)
- 2008 : contournement de la validation de certificats dans OpenSSL
- 2009 : collision MD5 sur des certificats concrets
- 2009 : attaque sur la renégociation
- 2009 : confusion liée à la présence de caractères nuls dans les certificats
- 2011 : BEAST (IV implicite dans le mode CBC)
- 2011 : Mauvaise interprétation de l'extension X.509 *Basic Constraints* (iOS)
- 2012 : *Mining your Ps and Qs* (absence d'aléa dans la génération de clé RSA)
- 2013 : *Lucky 13* (oracle de padding CBC) + biais statistiques sur RC4
- 2014 : `goto fail` Apple
- 2014 : contournement de la validation de certificats dans GnuTLS
- 2014 : *Triple Handshake* (renégociation et reprise de session)
- 2014 : *Heartbleed* et *EarlyCCS*
- 2015 : FREAK et *LogJam*
- 2016 : DROWN

Erreurs de conception dans la première phase de SSL/TLS

La première étape d'une connexion SSL/TLS est d'authentifier le serveur et d'accorder les parties sur un secret partagé

Quelques exemples d'échecs

Erreurs de conception dans la première phase de SSL/TLS

La première étape d'une connexion SSL/TLS est d'authentifier le serveur et d'accorder les parties sur un secret partagé

Quelques exemples d'échecs

- Négociation à la baisse dans SSLv2 : seules les valeurs aléatoires échangés étaient authentifiées

Erreurs de conception dans la première phase de SSL/TLS

La première étape d'une connexion SSL/TLS est d'authentifier le serveur et d'accorder les parties sur un secret partagé

Quelques exemples d'échecs

- Négociation à la baisse dans SSLv2 : seules les valeurs aléatoires échangés étaient authentifiées
- Vulnérabilité sur la renégociation / *Triple Handshake* : les différentes sessions/*époques* n'étaient pas liées de manière suffisante

Du *master secret* et de la dérivation des clés (1/2)

Le *master secret* est dérivé à partir des éléments suivants

- le résultat de l'algorithme d'échange de clé
- les valeurs aléatoires échangées en clair

L'intégrité n'est garantie qu'a posteriori, avec les messages `Finished`

Du *master secret* et de la dérivation des clés (1/2)

Le *master secret* est dérivé à partir des éléments suivants

- le résultat de l'algorithme d'échange de clé
- les valeurs aléatoires échangées en clair

L'intégrité n'est garantie qu'a posteriori, avec les messages `Finished`

Plusieurs attaques reposent sur cette couverture limitée dans la dérivation

- des attaques *cross-protocol* présentée dès 1999
- des instanciations (presque) pratiques publiées en 2012
- *Triple Handshake* joue dessus en 2014
- SMACK/FREAK en 2015

Du *master secret* et de la dérivation des clés (2/2)

Un correctif proposé

- l'extension *session-hash* : la dérivation de clé devrait couvrir tout les échanges précédents
- RFC 7627 pour TLS 1.{0,1,2}
- inclusion dans le nouveau standard TLS 1.3

Du *master secret* et de la dérivation des clés (2/2)

Un correctif proposé

- l'extension *session-hash* : la dérivation de clé devrait couvrir tout les échanges précédents
- RFC 7627 pour TLS 1.{0,1,2}
- inclusion dans le nouveau standard TLS 1.3

Cependant, ce n'est pas suffisant pour toutes les attaques, par exemple *LogJam*

- le groupe DH est choisi par le serveur en se basant sur des informations non authentifiées (on peut donc tomber sur un groupe faible de 512 bits)
- le groupe choisi est signé avec les valeurs aléatoires échangées
- un attaquant peut casser le logarithme discret sur le groupe en question et réutiliser ce message pour contrôler la communication

Du *master secret* et de la dérivation des clés (2/2)

Un correctif proposé

- l'extension *session-hash* : la dérivation de clé devrait couvrir tout les échanges précédents
- RFC 7627 pour TLS 1.{0,1,2}
- inclusion dans le nouveau standard TLS 1.3

Cependant, ce n'est pas suffisant pour toutes les attaques, par exemple *LogJam*

- le groupe DH est choisi par le serveur en se basant sur des informations non authentifiées (on peut donc tomber sur un groupe faible de 512 bits)
- le groupe choisi est signé avec les valeurs aléatoires échangées
- un attaquant peut casser le logarithme discret sur le groupe en question et réutiliser ce message pour contrôler la communication

Le véritable correctif : signer l'ensemble des messages précédents, et pas uniquement les valeurs aléatoires

Attaques affectant le *Record Protocol*

Depuis 2011, de nombreuses attaques pratiques contre le *Record Protocol* ont été publiées

- BEAST : exploitation de CBC avec un IV implicite (versions antérieures à TLS 1.1)
- CRIME (suivi de TIME et BREACH) : utilisation de la compression TLS/HTTP comme un canal auxiliaire
- Lucky 13 : oracle de padding dans le mode CBC
- RC4 : biais statistiques permettant de retrouver le clair
- POODLE : un oracle de padding plus efficace contre SSLv3

Attaques affectant le *Record Protocol*

Depuis 2011, de nombreuses attaques pratiques contre le *Record Protocol* ont été publiées

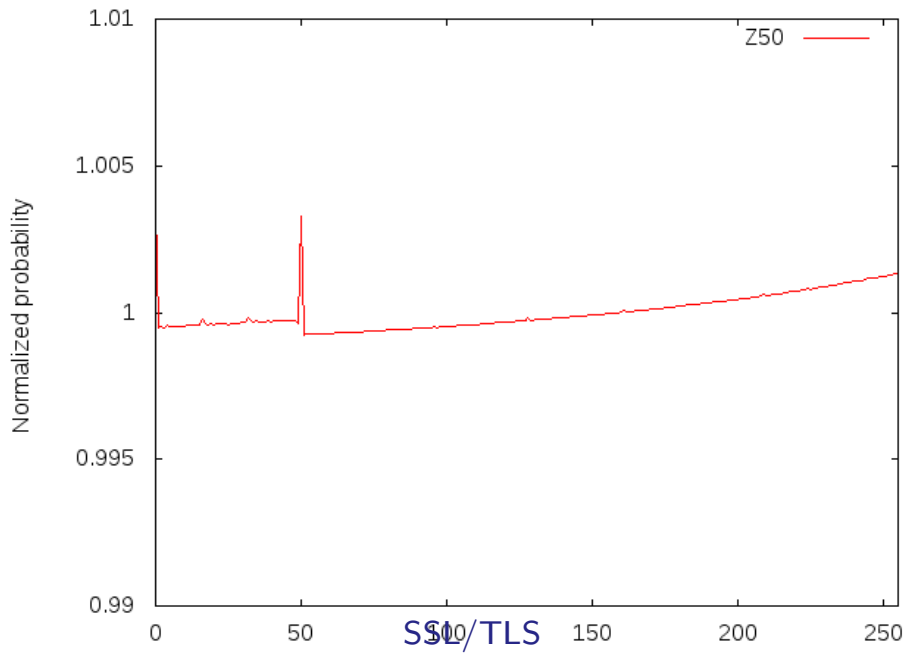
- BEAST : exploitation de CBC avec un IV implicite (versions antérieures à TLS 1.1)
- CRIME (suivi de TIME et BREACH) : utilisation de la compression TLS/HTTP comme un canal auxiliaire
- Lucky 13 : oracle de padding dans le mode CBC
- RC4 : biais statistiques permettant de retrouver le clair
- POODLE : un oracle de padding plus efficace contre SSLv3

Toutes ces attaques ciblent les cookies d'authentification HTTP

- en effet, ces cookies sont répétés dans différentes sessions TLS, rendant ces attaques possibles

Zoom sur RC4

- 1987 : RC4, un algorithme de chiffrement par flot propriétaire conçu par Rivest
- 1994 : RC4 publié officieusement sous le nom ARCFOUR
- 1995-2000 : divers biais sur les premiers octets de la suite chiffrante
- 2001 : WEP (qui utilise des clés corrélées) est cassé par Flührer et al.
- 2013 : des biais exploitables dans le cas de TLS sont mis au jour
- 2014 : amélioration des attaques



Réflexions autour du mode CBC

`https://www.openssl.org/~bodo/tls-cbc.txt`

Bodo Möller

Réflexions autour du mode CBC

`https://www.openssl.org/~bodo/tls-cbc.txt`

Bodo Möller

- Le mode CBC dans TLS favorise les oracles de padding
- L'utilisation d'un IV implicite peut mener à des attaques
- Avec SSLv3, les octets de padding sont mal spécifiés, ce qui amplifie les oracles

Réflexions autour du mode CBC

`https://www.openssl.org/~bodo/tls-cbc.txt`

Bodo Möller (2004-05-20)

- Le mode CBC dans TLS favorise les oracles de padding
- L'utilisation d'un IV implicite peut mener à des attaques
- Avec SSLv3, les octets de padding sont mal spécifiés, ce qui amplifie les oracles
- ... il décrivait Lucky 13, BEAST et POODLE

2014 : une année dure pour TLS

En 2014, toutes les piles TLS majeures ont été touchées par une vulnérabilité critique

- février : `goto fail` (Apple)
- février : `goto fail` (GnuTLS)
- avril : *Heartbleed* (OpenSSL)
- juin : *Early CCS* (OpenSSL)
- septembre : Universal signature forgery (Berserk ?) (Mozilla NSS)
- septembre : Universal signature forgery (Berserk ?) (CyaSSL)
- septembre : Universal signature forgery (Berserk ?) (mbedTLS)
- novembre : exécution de code arbitraire (MS SChannel)

2014 : une année dure pour TLS

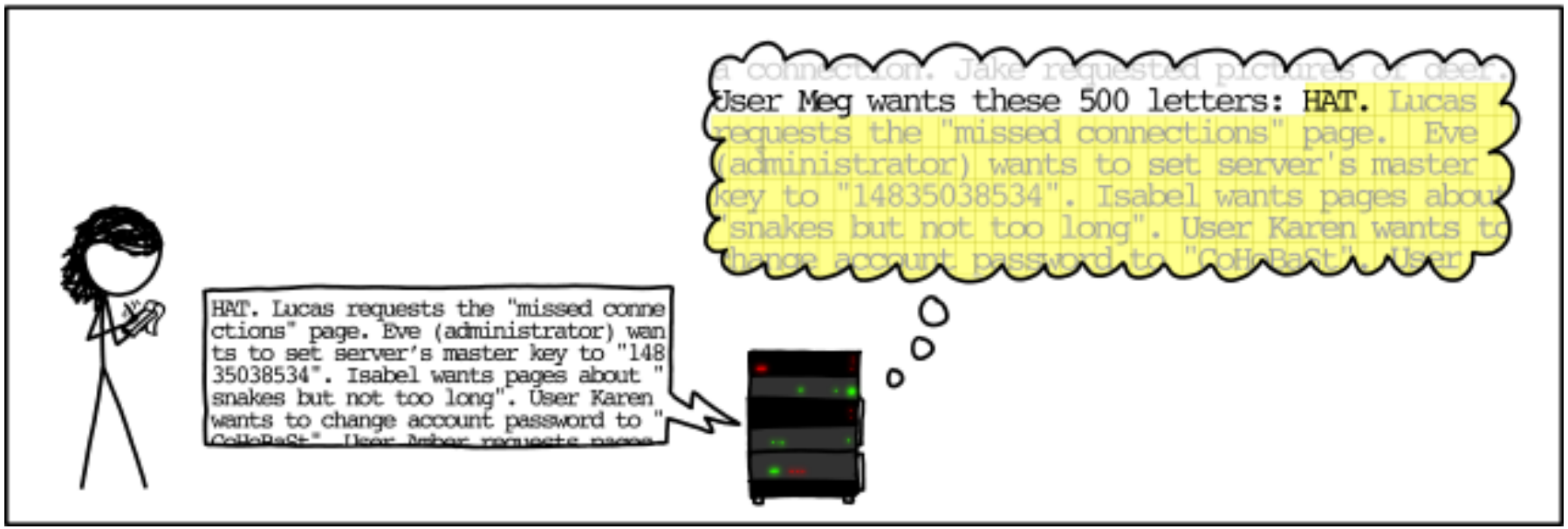
En 2014, toutes les piles TLS majeures ont été touchées par une vulnérabilité critique

- février : `goto fail` (Apple)
- février : `goto fail` (GnuTLS)
- avril : *Heartbleed* (OpenSSL)
- juin : *Early CCS* (OpenSSL)
- septembre : Universal signature forgery (Berserk ?) (Mozilla NSS)
- septembre : Universal signature forgery (Berserk ?) (CyaSSL)
- septembre : Universal signature forgery (Berserk ?) (mbedTLS)
- novembre : exécution de code arbitraire (MS SChannel)

Quizz

- Que s'est-il aussi passé le 8 avril 2014 (en plus de Heartbleed) ?
- Même question le 24 septembre 2014 (publication de Berserk) ?

Un débordement de tampon classique : Heartbleed



Source : `http://xkcd.com/1354`

```
+        /* Read type and payload length first */
+        if (1 + 2 + 16 > s->s3->rrec.length)
+                return 0; /* silently discard */
```

Un autre dépassement de tampon stupide : WinShock

- L'authentification client utilisant des certificats avec des courbes elliptiques repose sur deux messages
- Le message `Certificate`, qui contient la chaîne de certification
  - il contient la courbe utilisée
  - en particulier, il indique la taille L du corps sous-jacent
- `CertificateVerify`, qui contient une signature couvrant les messages précédents
  - la signature contient les coordonnées d'un point, d'une taille l
  - supposons que SChannel copie l octets dans une zone allouée pour L octets, sans se poser de question...

Un autre dépassement de tampon stupide : WinShock

- L'authentification client utilisant des certificats avec des courbes elliptiques repose sur deux messages
- Le message `Certificate`, qui contient la chaîne de certification
  - il contient la courbe utilisée
  - en particulier, il indique la taille L du corps sous-jacent
- `CertificateVerify`, qui contient une signature couvrant les messages précédents
  - la signature contient les coordonnées d'un point, d'une taille l
  - supposons que SChannel copie l octets dans une zone allouée pour L octets, sans se poser de question...

Sauf que l'authentification client par certificat est rarement utilisée

Un autre dépassement de tampon stupide : WinShock

- L'authentification client utilisant des certificats avec des courbes elliptiques repose sur deux messages
- Le message `Certificate`, qui contient la chaîne de certification
  - il contient la courbe utilisée
  - en particulier, il indique la taille L du corps sous-jacent
- `CertificateVerify`, qui contient une signature couvrant les messages précédents
  - la signature contient les coordonnées d'un point, d'une taille l
  - supposons que SChannel copie l octets dans une zone allouée pour L octets, sans se poser de question...

Sauf que l'authentification client par certificat est rarement utilisée

Teaser : tous les serveurs vulnérables étaient pourtant exploitables

goto fail Apple

```
/* Extract from Apple's sslKeyExchange.c */
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;
```

La syntaxe n'aide pas, mais on aimerait que le compilateur nous signale ce code qui est clairement mort

True, False, FILE_NOT_FOUND

Analysons la faille CVE-2014-0092 de GnuTLS (mars 2014) :

> *But this bug is arguably much worse than* Apple*'s, as it has allowed crafted certificates to evade validation check for all versions of* GnuTLS *ever released since that project got started in late 2000.[...]*
> *The* `check_if_ca` *function is supposed to return true (any non-zero value in C) or false (zero) depending on whether the issuer of the certificate is a certificate authority (CA). A true return should mean that the certificate passed muster and can be used further, but the bug meant that error returns were misinterpreted as certificate validations.*

True, False, FILE_NOT_FOUND

Analysons la faille CVE-2014-0092 de GnuTLS (mars 2014) :

> *But this bug is arguably much worse than* Apple*'s, as it has allowed crafted certificates to evade validation check for all versions of* GnuTLS *ever released since that project got started in late 2000.[...]*
> *The* `check_if_ca` *function is supposed to return true (any non-zero value in C) or false (zero) depending on whether the issuer of the certificate is a certificate authority (CA). A true return should mean that the certificate passed muster and can be used further, but the bug meant that error returns were misinterpreted as certificate validations.*

Au passage, une faille similaire avait été trouvée dans OpenSSL... en 2008 (CVE-2008-5077).

L'oubli de l'extension `Basic constraints`

Un *bug* de Microsoft Internet Explorer découvert par Marlinspike en 2002 :

- la pile X.509 ne vérifiait pas l'extension `Basic Constraints`
- tout certificat final pouvait servir de certificat d'autorité

L'oubli de l'extension `Basic constraints`

Un *bug* de Microsoft Internet Explorer découvert par Marlinspike en 2002 :

- la pile X.509 ne vérifiait pas l'extension `Basic Constraints`
- tout certificat final pouvait servir de certificat d'autorité

Le même *bug*, en 2010, dans Apple iOS...

L'oubli de l'extension `Basic constraints`

Un *bug* de Microsoft Internet Explorer découvert par Marlinspike en 2002 :

- la pile X.509 ne vérifiait pas l'extension `Basic Constraints`
- tout certificat final pouvait servir de certificat d'autorité

Le même *bug*, en 2010, dans Apple iOS...

Il est peut-être temps de considérer que les développeurs ne sont pas seuls responsables... et d'améliorer nos langages/outils/spécifications

Bleichenbacher (1/2)

RSA PKCS#1 v1.5

- le chiffrement RSA repose sur un schéma de bourrage
- comment traiter un *padding* invalide lors du déchiffrement ?

Bleichenbacher (1/2)

RSA PKCS#1 v1.5

- le chiffrement RSA repose sur un schéma de bourrage
- comment traiter un *padding* invalide lors du déchiffrement ?

L'attaque de Bleichenbacher (1998)

- le principe : envoyer un texte chiffré altéré
- si l'attaquant peut distinguer un *padding* valide d'un *padding* invalide, il obtient de l'information sur le clair
- application à TLS : l'attaque *Million Message Attack*

Bleichenbacher (2/2)

L'attaque resurgit en 2014

- en Java, une erreur de *padding* provoque une exception
- ainsi, pour éviter une *timing attack*, il faut redévelopper l'algorithme
- un développeur TLS doit choisir entre la modularité et la sécurité

Bleichenbacher (2/2)

L'attaque resurgit en 2014

- en Java, une erreur de *padding* provoque une exception
- ainsi, pour éviter une *timing attack*, il faut redévelopper l'algorithme
- un développeur TLS doit choisir entre la modularité et la sécurité

Et en 2016...

- DROWN *(Decrypting RSA with Obsolete and Weakened eNcryption)*
- attaque sur SSLv2 pour retrouver des *pre-master secret* TLS
- des *bugs* dans OpenSSL rendent l'attaque très efficace

Bleichenbacher (2/2)

L'attaque resurgit en 2014

- en Java, une erreur de *padding* provoque une exception
- ainsi, pour éviter une *timing attack*, il faut redévelopper l'algorithme
- un développeur TLS doit choisir entre la modularité et la sécurité

Et en 2016...

- DROWN *(Decrypting RSA with Obsolete and Weakened eNcryption)*
- attaque sur SSLv2 pour retrouver des *pre-master secret* TLS
- des *bugs* dans OpenSSL rendent l'attaque très efficace
- heureusement que SSLv2 est considéré comme obsolète depuis plus de 10 ans...

MAC-then-Encrypt

Les oracles de *padding* existent aussi en symétrique

- *MAC-then-CBC* est naturellement vulnérable
- 2002 : Vaudenay présente le principe de l'attaque
- 2011 : *XML Encryption is broken*
- 2013 : Lucky 13 (l'attaque est applicable à TLS)

MAC-then-Encrypt

Les oracles de *padding* existent aussi en symétrique

- *MAC-then-CBC* est naturellement vulnérable
- 2002 : Vaudenay présente le principe de l'attaque
- 2011 : *XML Encryption is broken*
- 2013 : Lucky 13 (l'attaque est applicable à TLS)

Le correctif ?

- du sparadrap (une note d'implémentation dans le standard)
- un *patch* sordide pour garantir un déchiffrement en temps constant
- utiliser `Encrypt-then-MAC` (RFC 7366) ou du chiffrement authentifié (AEAD)

Là encore, il faut choisir entre modularité et sécurité (et interopérabilité)

SMACK et FREAK

En 2015, des attaques concernant presque toutes les piles TLS

- En Java, l'envoi d'un message `Finished` précoce permettait à un attaquant de sauter toute la négociation (dont l'authentification)

SMACK et FREAK

En 2015, des attaques concernant presque toutes les piles TLS

- En Java, l'envoi d'un message `Finished` précoce permettait à un attaquant de sauter toute la négociation (dont l'authentification)
- Dans certaines attaques, il y avait confusion entre les points utilisés par ECDHE et ECDSA lorsque certains messages sont supprimés

SMACK et FREAK

En 2015, des attaques concernant presque toutes les piles TLS

- En Java, l'envoi d'un message `Finished` précoce permettait à un attaquant de sauter toute la négociation (dont l'authentification)
- Dans certaines attaques, il y avait confusion entre les points utilisés par ECDHE et ECDSA lorsque certains messages sont supprimés
- OpenSSL acceptait l'échange de clé RSA-EXPORT lorsque le chiffrement RSA a été négocié (FREAK)

SMACK et FREAK

En 2015, des attaques concernant presque toutes les piles TLS

- En Java, l'envoi d'un message `Finished` précoce permettait à un attaquant de sauter toute la négociation (dont l'authentification)
- Dans certaines attaques, il y avait confusion entre les points utilisés par ECDHE et ECDSA lorsque certains messages sont supprimés
- OpenSSL acceptait l'échange de clé RSA-EXPORT lorsque le chiffrement RSA a été négocié (FREAK)

En général, la machine à états réagit aux messages reçus, au lieu de maintenir une liste restreinte des messages licites

Autres problèmes des machines à états

- *Early CCS*, trouvé à l'aide de méthodes formelles
- Dans la vulnérabilité SChannel précédente, les messages `Certificate` et `CertificateVerify` étaient toujours interprétés, même s'ils n'étaient pas sollicités

Presque toutes les piles TLS étaient vulnérables à des attaques similaires :

- Les specs sont peut-être trop complexes ?
- Nous avons sans doute besoin de plus de tests

TLS 1.3 : un nouvel espoir

Parmi les failles présentées, nombreuses sont celles issues de la conception

- Crypto obsolète, retirée dans TLS 1.3 : PKCS#1 v1.5, RC4, CBC...
- Problèmes de l'automate : la négociation a été repensée
- En parallèle des efforts de spécification, le protocole fait l'objet de modélisation et de tests en utilisant des méthodes formelles (TRON workshop en 2016)

Cependant, TLS 1.3 ne résout pas

- les soucis de compatibilité : les versions précédentes de TLS sont encore là pour un certain temps
- le standard introduit de nouvelles formes de complexité
  - 0-RTT
  - L'authentification client et le rafraîchissement des clés encore mouvants

Langages et méthodologie

Nos langages de programmation devraient nous aider

- le typage fort permet d'éviter des erreurs simples
- la gestion de la mémoire peut être sûre

De bons outils sont nécessaires

- il faut activer les *warnings*
- et les corriger (`-Werror`)

Nous avons besoin de tests

- tests de non-régression
- tests négatifs

Questions

Merci pour votre attention

olivier.levillain@ssi.gouv.fr