Influence de la qualité des spécifications sur la sécurité logicielle

Olivier Levillain

Séminaire SCN du 11 octobre 2022

Plan

Mise en jambes : Mini-PNG

PDF : la complexité et la tolérance au service de l'insécurité

Heartbleed vu de la RFC 6520

Les machines dans tous leurs états

Quelques éléments de conclusion

Plan

Description du contexte

Exercice pratique noté dans un module « Programmation orientée sécurité »

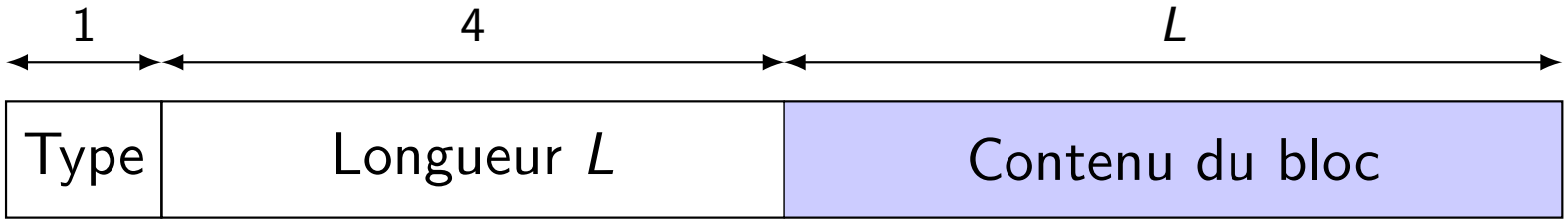
- spécification d'un format d'images simple inspiré de PNG
- échantillons d'images labellisées (correctes et incorrectes)
- objectifs de l'exercice
  - développer un *parser* pour ce format
  - produire du code « sécurisé »
  - proposer un regard critique sur la spécification

Le cours « POS » a été présenté plus en détails dans [RESSI18]

Mini-PNG en une planche

Structure du fichier

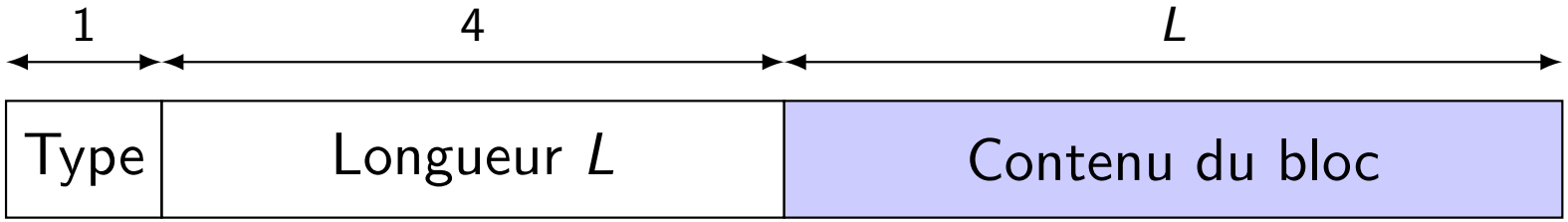
- un marqueur « `Mini-PNG` »
- une suite de blocs au format suivant :



Mini-PNG en une planche

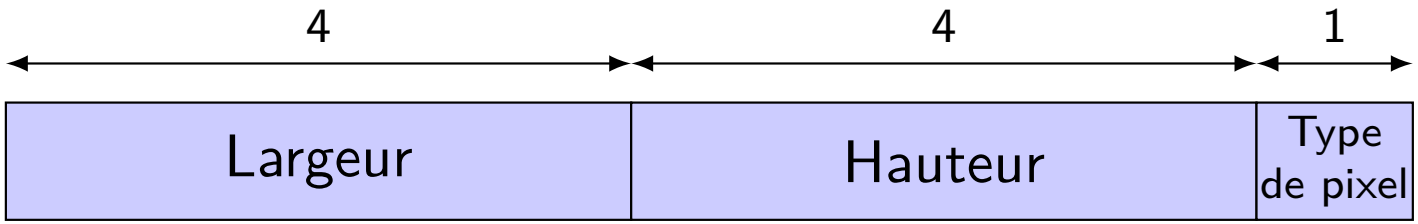
Structure du fichier

- un marqueur « `Mini-PNG` »
- une suite de blocs au format suivant :



D'après la spécification, un fichier PNG doit contenir

- un bloc H (en-tête), dont le contenu est le suivant :



- 0, 1 ou plusieurs blocs C (commentaire)
- au moins 1 bloc D (données)

Ambiguïtés et hypothèses implicites

L'ordre des blocs n'est pas vraiment explicité

- les étudiants adaptent donc leur code aux échantillons
- leurs *parsers* font alors des hypothèses implicites
- parfois, les positions des différents blocs sont même fixées dans le code

Ambiguïtés et hypothèses implicites

L'ordre des blocs n'est pas vraiment explicité

- les étudiants adaptent donc leur code aux échantillons
- leurs *parsers* font alors des hypothèses implicites
- parfois, les positions des différents blocs sont même fixées dans le code

Conséquences sur le fonctionnement

- rejet de fichiers bien formés
- divers problèmes de gestion mémoire, par exemple
  - allocation mémoire réalisée lors de la lecture du bloc H
  - copie des données à l'interprétation du bloc D
  - hypothèse implicite pas toujours vérifiée : H avant D

Encodage des entiers

L'encodage des entiers n'est pas spécifié dans la spécification

- on déduit de l'examen des échantillons que les entiers sont *big endian*
- peu d'élèves s'émeuvent de l'absence de cette information

Encodage des entiers

L'encodage des entiers n'est pas spécifié dans la spécification

- on déduit de l'examen des échantillons que les entiers sont *big endian*
- peu d'élèves s'émeuvent de l'absence de cette information

Il va sans dire que l'ensemble des entiers sont positifs

- cette précision est également absente des spécifications
- interpréter une longueur de bloc comme un entier négatif peut avoir un impact fâcheux

Encodage des entiers

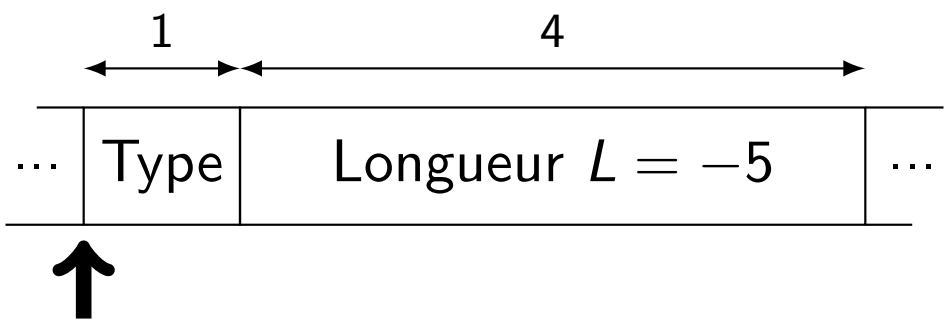
L'encodage des entiers n'est pas spécifié dans la spécification

- on déduit de l'examen des échantillons que les entiers sont *big endian*
- peu d'élèves s'émeuvent de l'absence de cette information

Il va sans dire que l'ensemble des entiers sont positifs

- cette précision est également absente des spécifications
- interpréter une longueur de bloc comme un entier négatif peut avoir un impact fâcheux

Exemple de code réel : analyse du fichier en mémoire via un tableau



Encodage des entiers

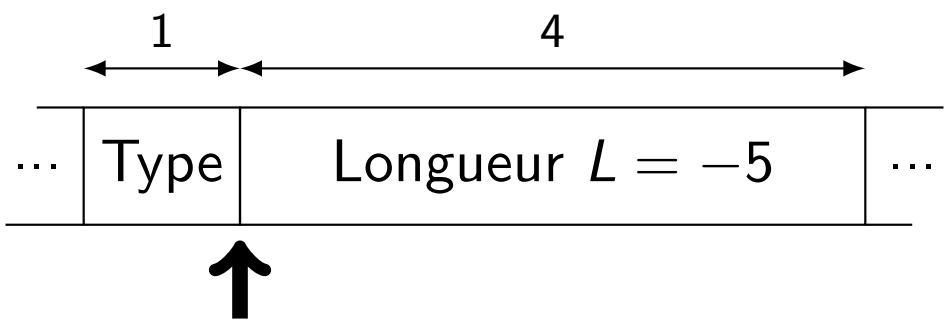
L'encodage des entiers n'est pas spécifié dans la spécification

- on déduit de l'examen des échantillons que les entiers sont *big endian*
- peu d'élèves s'émeuvent de l'absence de cette information

Il va sans dire que l'ensemble des entiers sont positifs

- cette précision est également absente des spécifications
- interpréter une longueur de bloc comme un entier négatif peut avoir un impact fâcheux

Exemple de code réel : analyse du fichier en mémoire via un tableau



Encodage des entiers

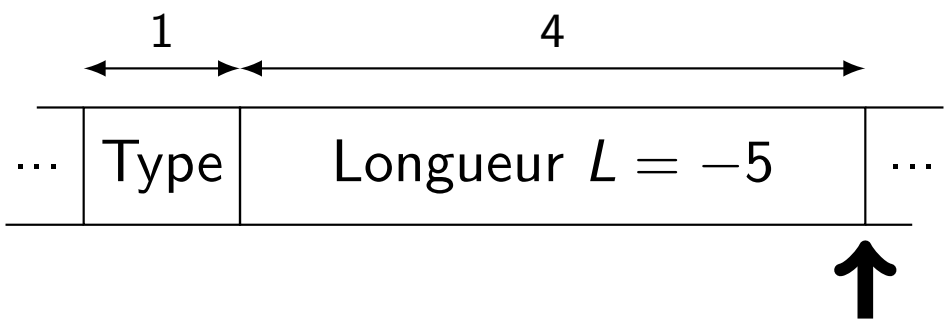
L'encodage des entiers n'est pas spécifié dans la spécification

- on déduit de l'examen des échantillons que les entiers sont *big endian*
- peu d'élèves s'émeuvent de l'absence de cette information

Il va sans dire que l'ensemble des entiers sont positifs

- cette précision est également absente des spécifications
- interpréter une longueur de bloc comme un entier négatif peut avoir un impact fâcheux

Exemple de code réel : analyse du fichier en mémoire via un tableau



Encodage des entiers

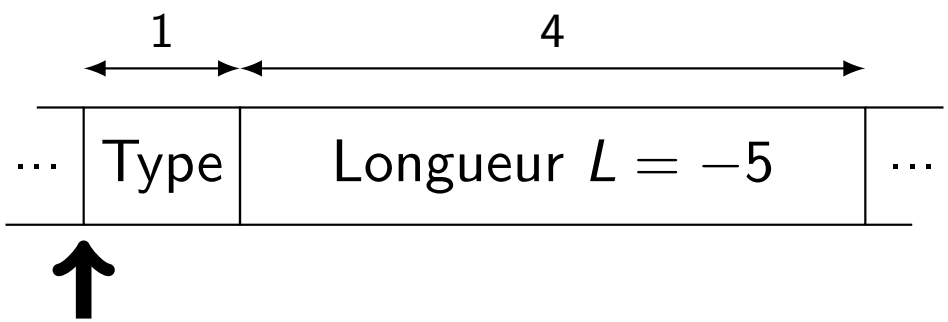
L'encodage des entiers n'est pas spécifié dans la spécification

- on déduit de l'examen des échantillons que les entiers sont *big endian*
- peu d'élèves s'émeuvent de l'absence de cette information

Il va sans dire que l'ensemble des entiers sont positifs

- cette précision est également absente des spécifications
- interpréter une longueur de bloc comme un entier négatif peut avoir un impact fâcheux

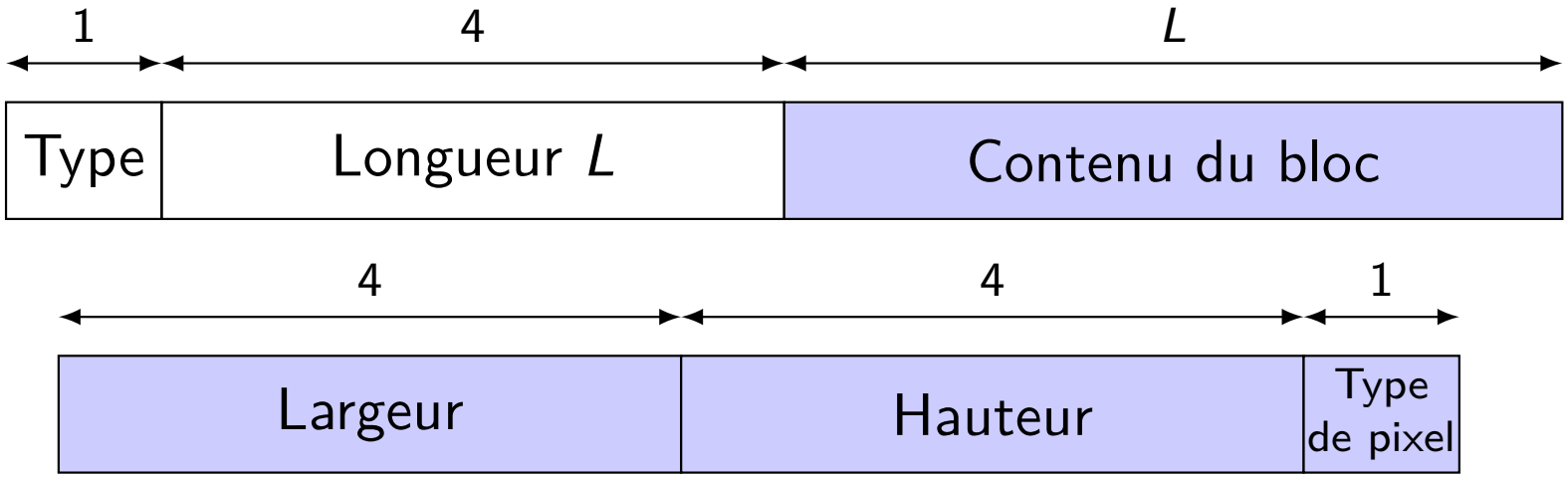
Exemple de code réel : analyse du fichier en mémoire via un tableau



Ambiguïtés et interprétations concurrentes

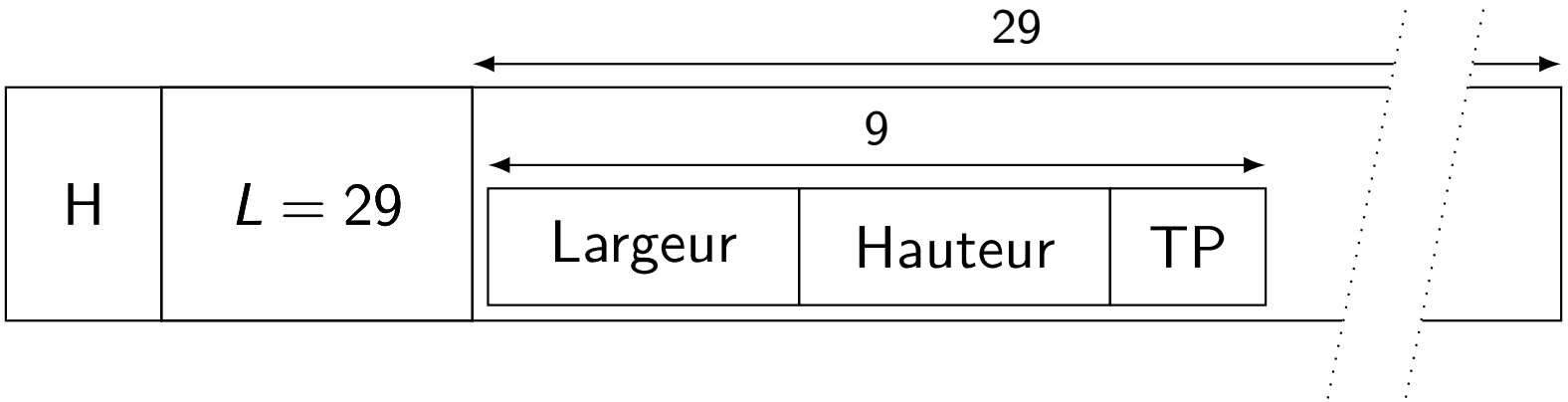
La longueur d'un bloc H est « définie » à deux endroits dans la spécification

- longueur explicite du bloc (L)
- longueur de la structure incluse (qui vaut 9)



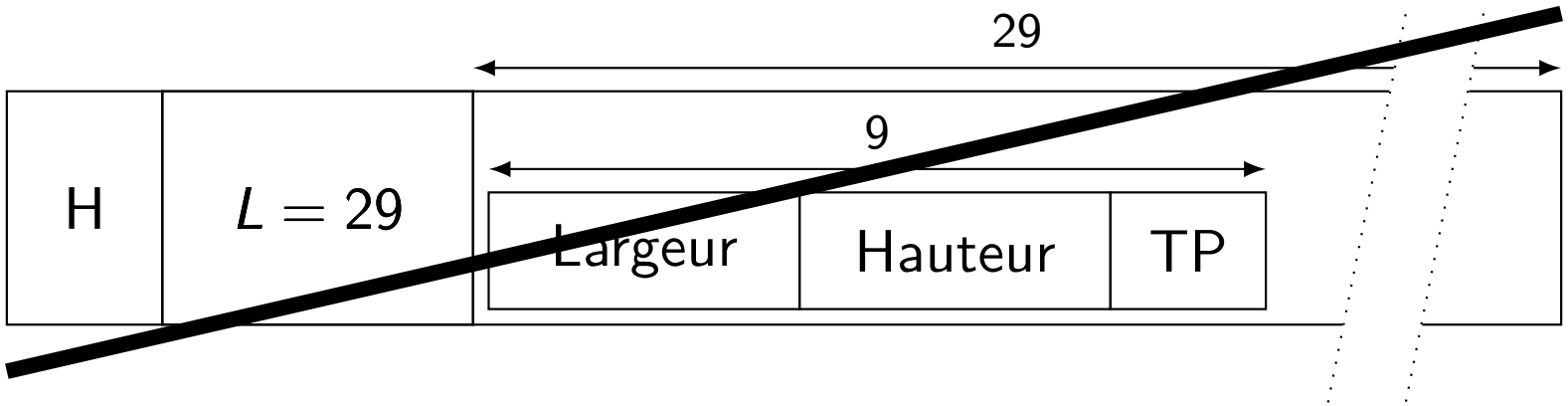
Que se passe-t-il quand L ne vaut pas 9 ?

Ambiguïtés et interprétations concurrentes



En l'absence d'instructions claires dans la spécification, il existe différentes interprétations

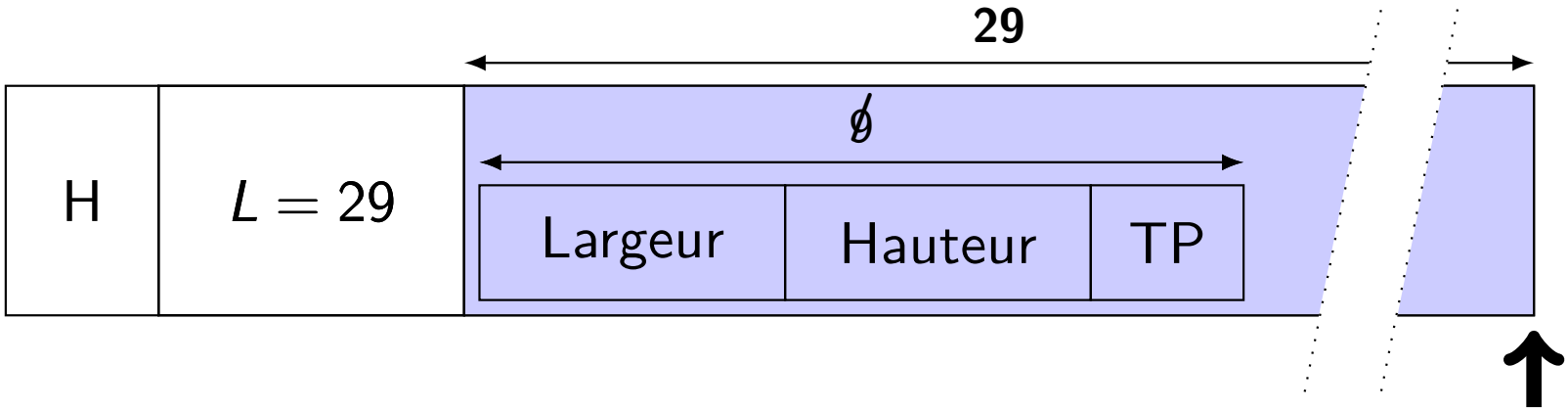
Ambiguïtés et interprétations concurrentes



En l'absence d'instructions claires dans la spécification, il existe différentes interprétations

- le bloc est invalide

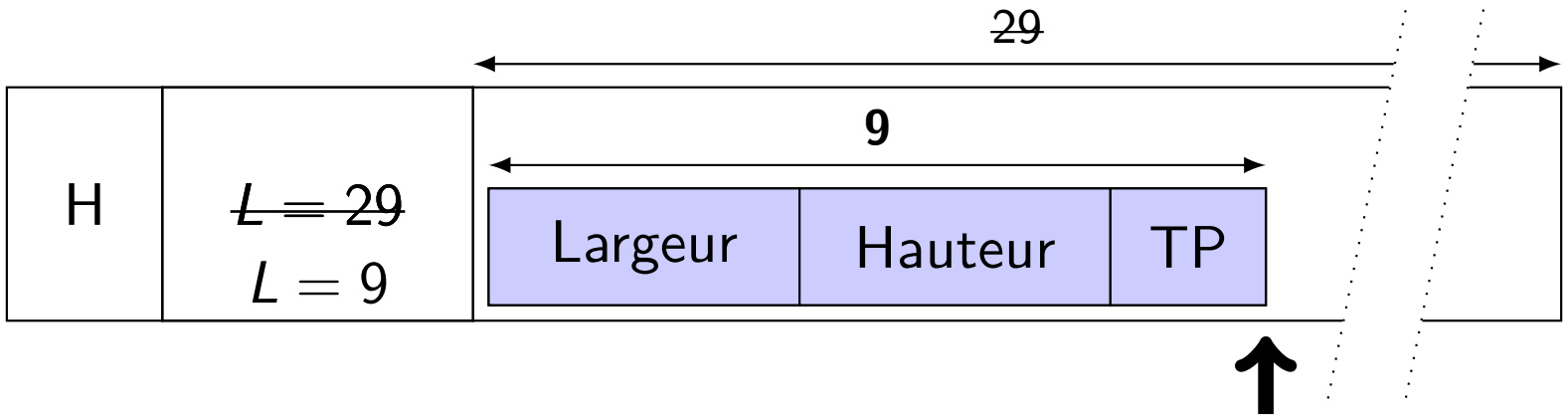
Ambiguïtés et interprétations concurrentes



En l'absence d'instructions claires dans la spécification, il existe différentes interprétations

- le bloc est invalide
- on ignore les octets en trop à la fin du bloc

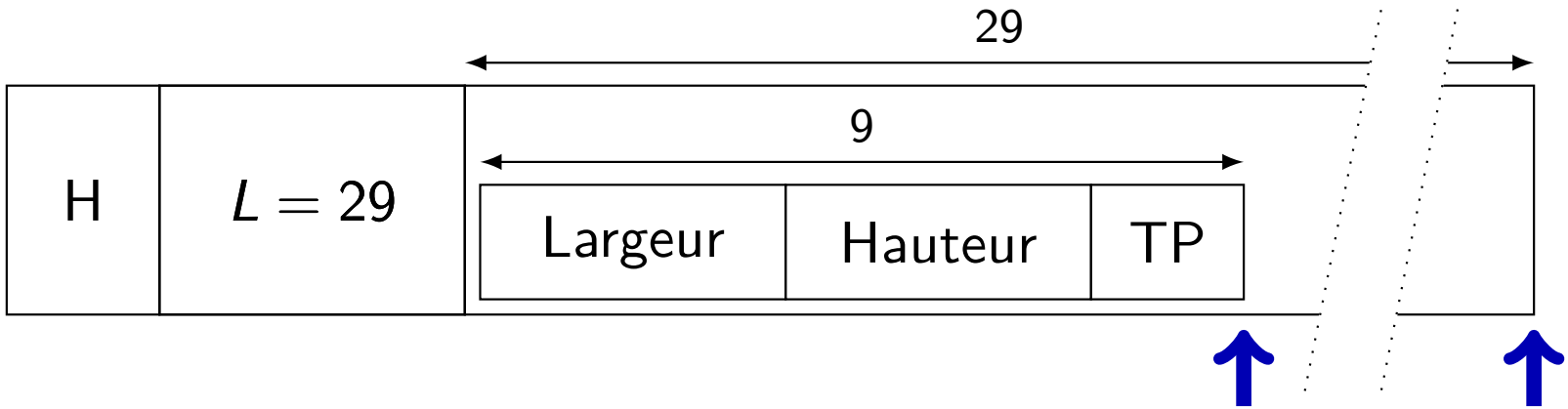
Ambiguïtés et interprétations concurrentes



En l'absence d'instructions claires dans la spécification, il existe différentes interprétations

- le bloc est invalide
- on ignore les octets en trop à la fin du bloc
- on ignore la longueur du bloc puisqu'on *sait* que l'en-tête fait 9 octets

Ambiguïtés et interprétations concurrentes



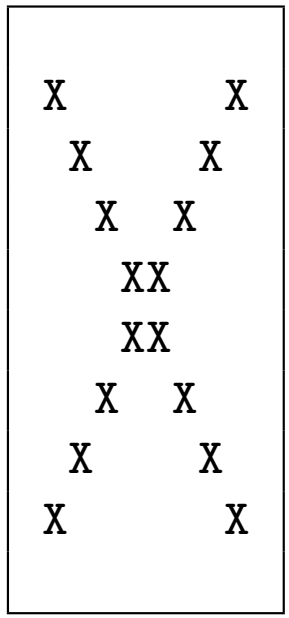
En l'absence d'instructions claires dans la spécification, il existe différentes interprétations

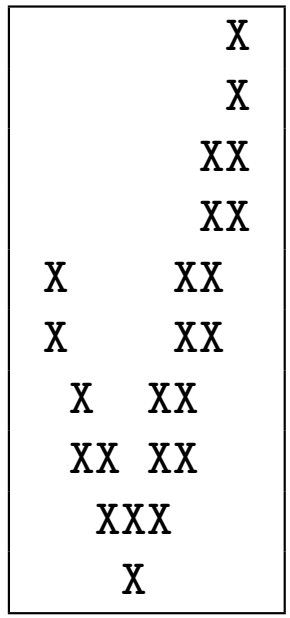
- le bloc est invalide
- on ignore les octets en trop à la fin du bloc
- on ignore la longueur du bloc puisqu'on *sait* que l'en-tête fait 9 octets

Ambiguïtés et interprétations concurrentes

Une image contenant un tel bloc H peut s'afficher différemment

`Invalid Header`
Pas d'affichage





Le principe de robustesse comme guide naturel

Postel : « Be liberal in what you accept, be strict in what you send »

- principe jugé nécessaire pour le développement et l'interopérabilité entre piles TCP/IP
- état d'esprit dangereux pour la sécurité
- penchant naturel des élèves à tout faire pour que « ça marche »

Le principe de robustesse comme guide naturel

Postel : « Be liberal in what you accept, be strict in what you send »

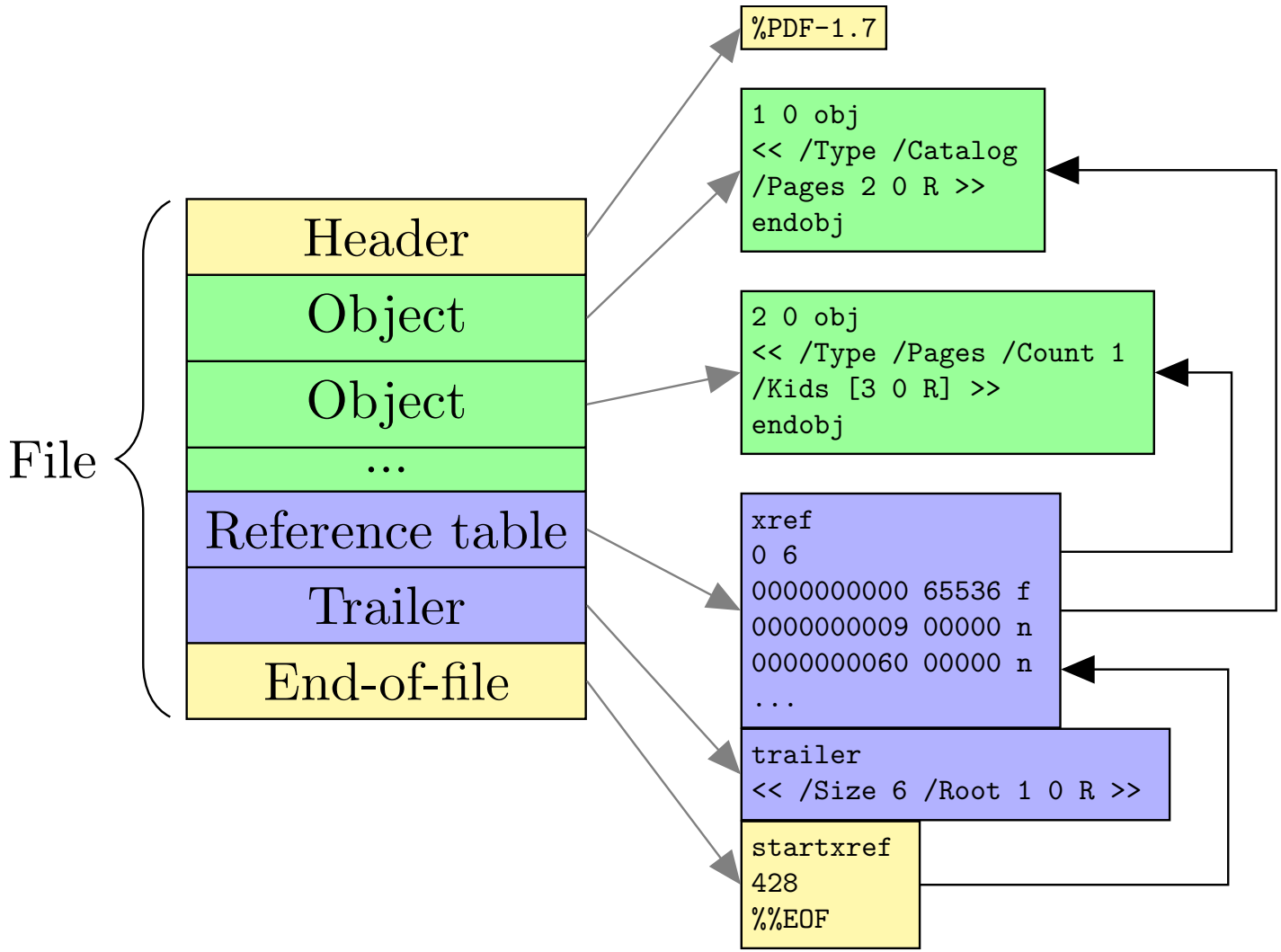
- principe jugé nécessaire pour le développement et l'interopérabilité entre piles TCP/IP
- état d'esprit dangereux pour la sécurité
- penchant naturel des élèves à tout faire pour que « ça marche »

Enseignements à tirer ?

- il faut semer les graines tôt pour cultiver un état d'esprit de sécurité
- les spécifications devraient être claires, complètes et explicites

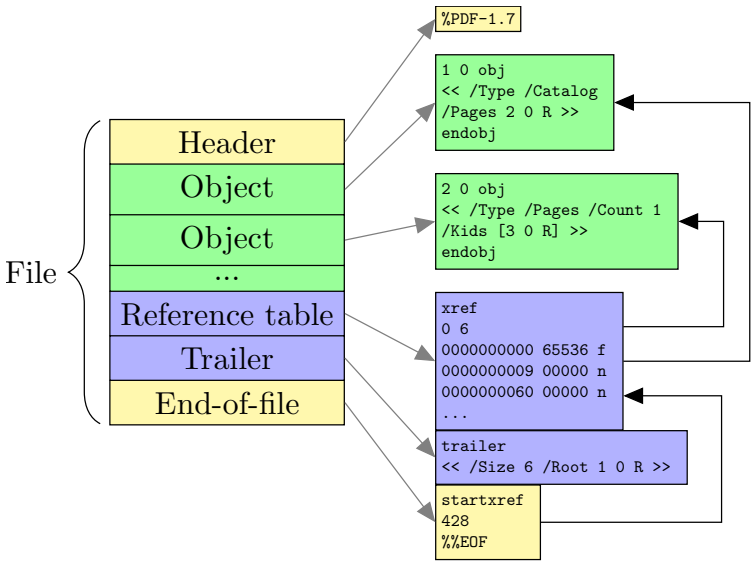
Plan

Un PDF minimaliste



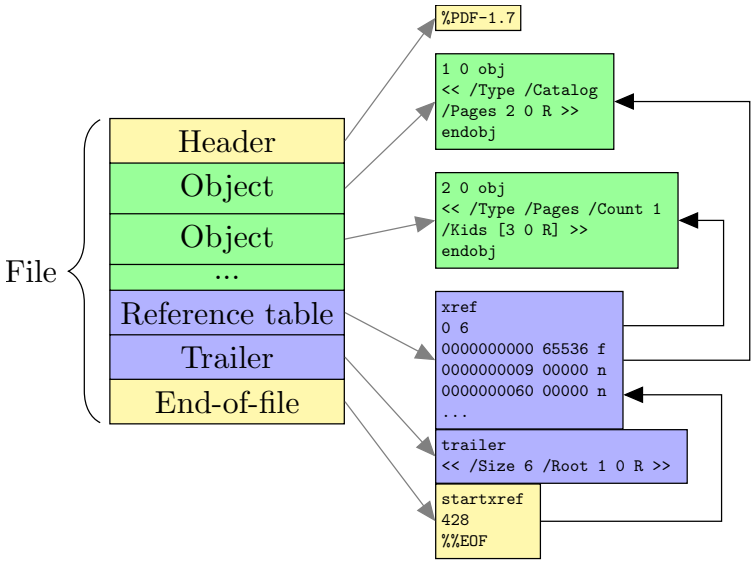
Cette section décrit des travaux réalisés avec G. Endignoux et J.-Y. Migeon [LangSec17]

Comment *parser* un PDF ?



- Vérification du marqueur %PDF
- Lecture de l'adresse de la table *xref*
- Interprétation des objets (via les offsets)
- Identification de la racine du document dans le *trailer*
- Parcours de l'arbre du document

Comment *parser* un PDF ?



- Vérification du marqueur %PDF
- Lecture de l'adresse de la table *xref*
- Interprétation des objets (via les offsets)
- Identification de la racine du document dans le *trailer*
- Parcours de l'arbre du document

Remarques

- processus complexe qui oblige à explorer le fichier via des *pointeurs*
- format à la fois textuel et binaire
- possibilité de modifier le document en *ajoutant* des objets
- PDF a été optimisé pour la mise à jour *et non pour la lecture*

Aperçu de la complexité de la structure d'un PDF

De nombreuses fonctionnalités viennent compléter le tableau

- mises à jour incrémentales
- compression des objets
- compression des tables *xref*
- fichiers *linéarisés*
- chiffrement et signature

Aperçu de la complexité de la structure d'un PDF

De nombreuses fonctionnalités viennent compléter le tableau

- mises à jour incrémentales
- compression des objets
- compression des tables *xref*
- fichiers *linéarisés*
- chiffrement et signature

Conséquence : nombreuses vulnérabilités sur les lecteurs classiques

- corruptions mémoire
- dénis de service
- exécution de code arbitraire

Aperçu de la complexité de la structure d'un PDF

De nombreuses fonctionnalités viennent compléter le tableau

- mises à jour incrémentales
- compression des objets
- compression des tables *xref*
- fichiers *linéarisés*
- chiffrement et signature

Conséquence : nombreuses vulnérabilités sur les lecteurs classiques

- corruptions mémoire
- dénis de service
- exécution de code arbitraire

Et il n'est question ici que de structure du fichier...

Écarts à la spécification

Tolérances quant au marqueur `%PDF`

- pour certains lecteurs, il peut être absent
- pour d'autres, il suffit qu'il soit *plutôt* au début

Écarts à la spécification

Tolérances quant au marqueur `%PDF`

- pour certains lecteurs, il peut être absent
- pour d'autres, il suffit qu'il soit *plutôt* au début

Réactions variées aux erreurs de syntaxe

- caractères invalides dans les clés de dictionnaires
- erreurs de typage menant à une confusion
- reconstruction des offsets dans un fichier corrompu

Écarts à la spécification

Tolérances quant au marqueur `%PDF`

- pour certains lecteurs, il peut être absent
- pour d'autres, il suffit qu'il soit *plutôt* au début

Réactions variées aux erreurs de syntaxe

- caractères invalides dans les clés de dictionnaires
- erreurs de typage menant à une confusion
- reconstruction des offsets dans un fichier corrompu

Certains de ces comportements sont encouragés par la spécification

> *When a conforming reader reads a PDF file with a damaged or missing cross-reference table, it **may attempt** to rebuild the table by scanning all the objects in the file.*
>
> — ISO 32000-1 2008 Annexe C.2

Conséquences sur la sécurité

Fichiers polyglottes

- possibilité de créer un fichier aux formats multiples
- travaux d'Ange Albertini (`https://corkami.github.io/`)
- confusion permettant de contourner certains mécanismes de sécurité

Conséquences sur la sécurité

Fichiers polyglottes

- possibilité de créer un fichier aux formats multiples
- travaux d'Ange Albertini (`https://corkami.github.io/`)
- confusion permettant de contourner certains mécanismes de sécurité

Absence d'affichage de confiance

- deux lecteurs peuvent afficher différemment le même fichier
- problème gênant lorsque le fichier est signé...

Conséquences sur la sécurité

Fichiers polyglottes

- possibilité de créer un fichier aux formats multiples
- travaux d'Ange Albertini (`https://corkami.github.io/`)
- confusion permettant de contourner certains mécanismes de sécurité

Absence d'affichage de confiance

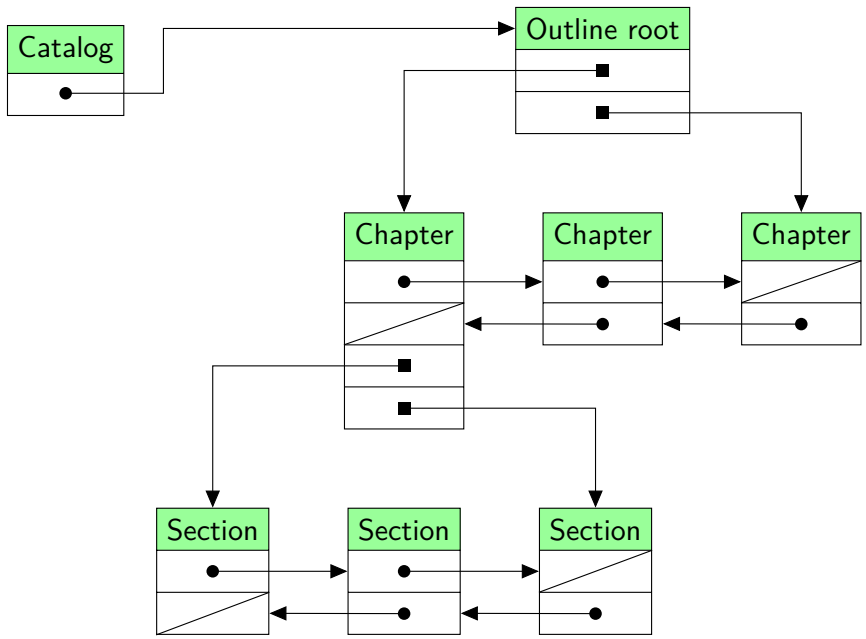
- deux lecteurs peuvent afficher différemment le même fichier
- problème gênant lorsque le fichier est signé...

Marge de manœuvre pour des attaques sophistiquées

- collisions SHA-1 de fichiers PDF
- Stevens et Al. – SHAttered, CRYPTO 2017

Utilisation de représentations inadaptées

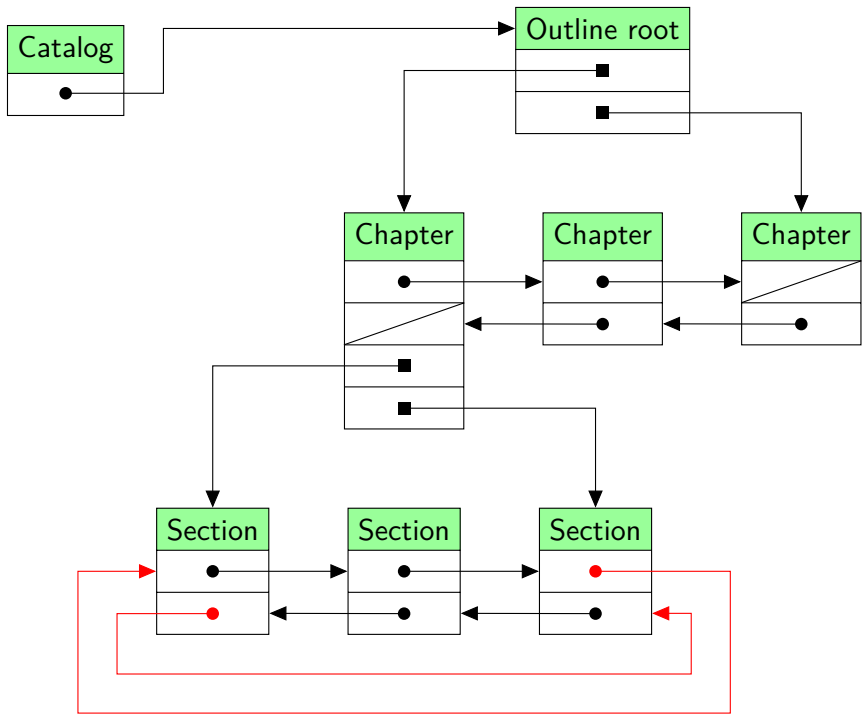
Pour décrire la table des matières, PDF utilise des listes doublement chaînées



Cette structure de données est plus expressive qu'un arbre

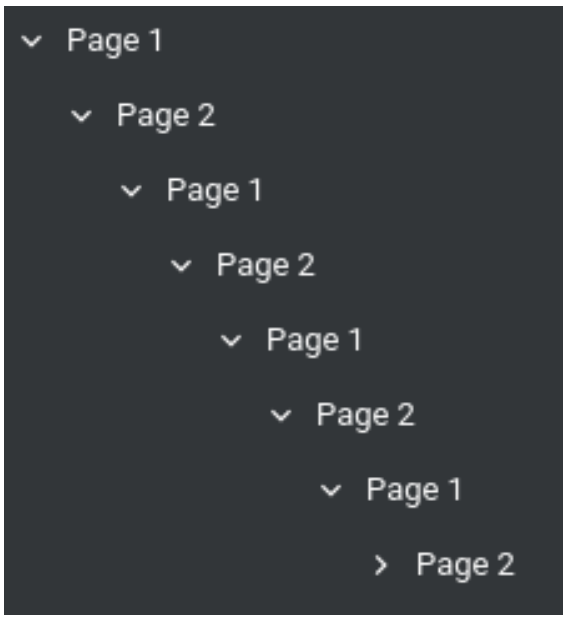
Utilisation de représentations inadaptées

Pour décrire la table des matières, PDF utilise des listes doublement chaînées



Que se passe-t-il si on ajoute des cycles ?

Utilisation de représentations inadaptées



Comportements identifiés

- déni de service (plantage ou conso. CPU)
- erreurs dans l'affichage (e.g. Chromium)
- rejet du fichier

Conclusion sur PDF

La complexité et l'ambiguïté sont des ennemis de la sécurité

- PDF est défini par un grand corpus de documents
- la spécification et l'écosystème encouragent les lecteurs à être accommodants face à des fichiers malformés

Conclusion sur PDF

La complexité et l'ambiguïté sont des ennemis de la sécurité

- PDF est défini par un grand corpus de documents
- la spécification et l'écosystème encouragent les lecteurs à être accommodants face à des fichiers malformés

Problèmes de sécurité proches de ceux de Mini-PNG

- corruptions mémoire
- interprétations différentes d'un même fichier
- erreurs de logique menant à des boucles infinies

Conclusion sur PDF

La complexité et l'ambiguïté sont des ennemis de la sécurité

- PDF est défini par un grand corpus de documents
- la spécification et l'écosystème encouragent les lecteurs à être accommodants face à des fichiers malformés

Problèmes de sécurité proches de ceux de Mini-PNG

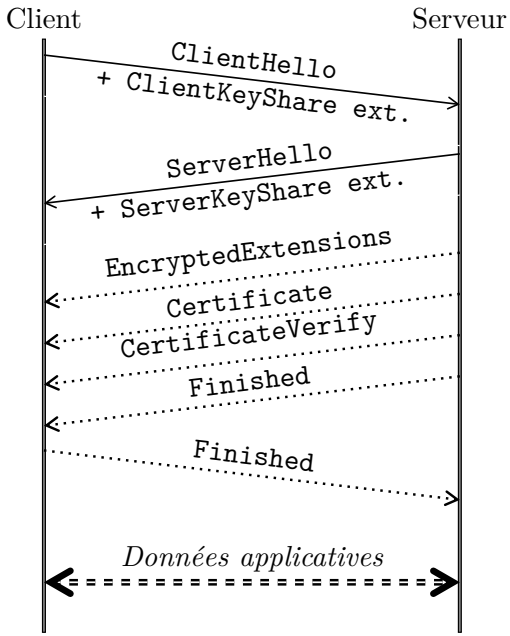
- corruptions mémoire
- interprétations différentes d'un même fichier
- erreurs de logique menant à des boucles infinies

Solutions ?

- Formaliser PDF à l'aide d'une grammaire classique, plus restrictive (BNF)
- Remplacer la loi de Postel par une règle plus simple : « *Be strict* »

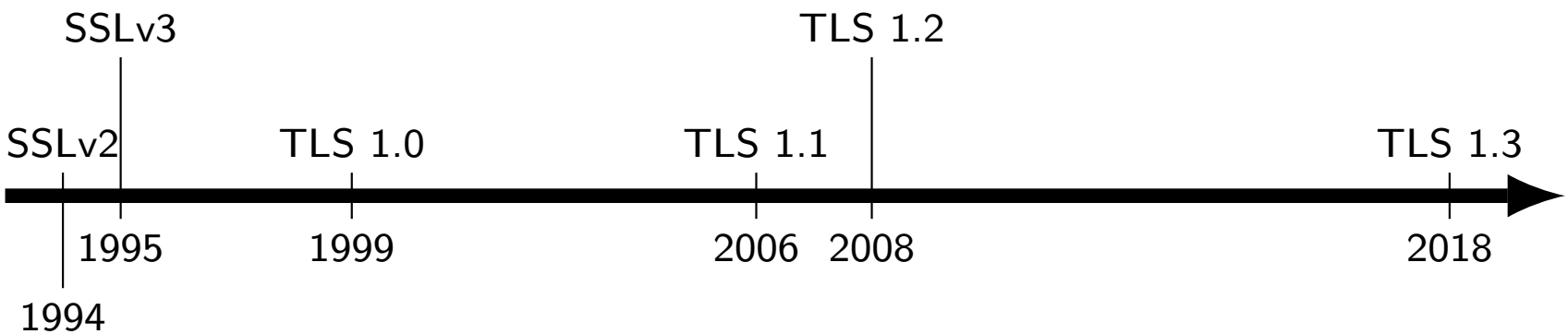
Plan

TLS en une planche



Objectifs de TLS

- authentifier le serveur
- établir un secret partagé
- protéger les données en confidentialité et en intégrité



Plus d'information sur TLS dans [PhD16] et [CRiSIS20]

Retour sur *Heartbleed* : les *records*

CVE-2014-0160

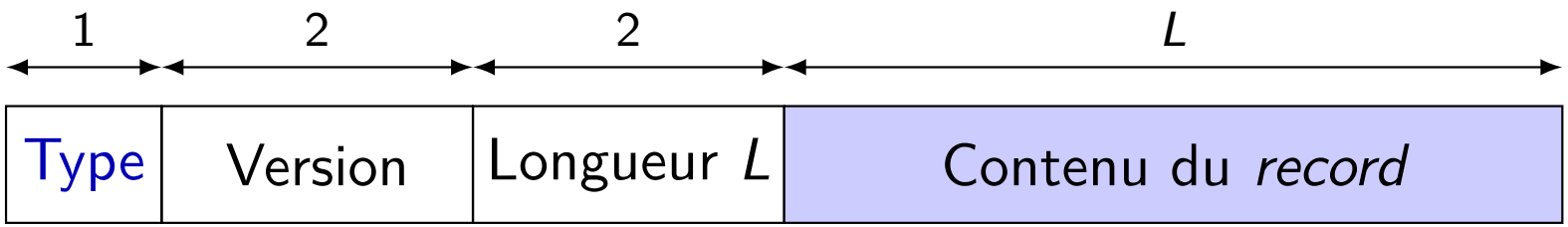
- divulgation d'information dans OpenSSL
- cause : lecture au-delà de la fin d'un tableau

Retour sur *Heartbleed* : les *records*

CVE-2014-0160

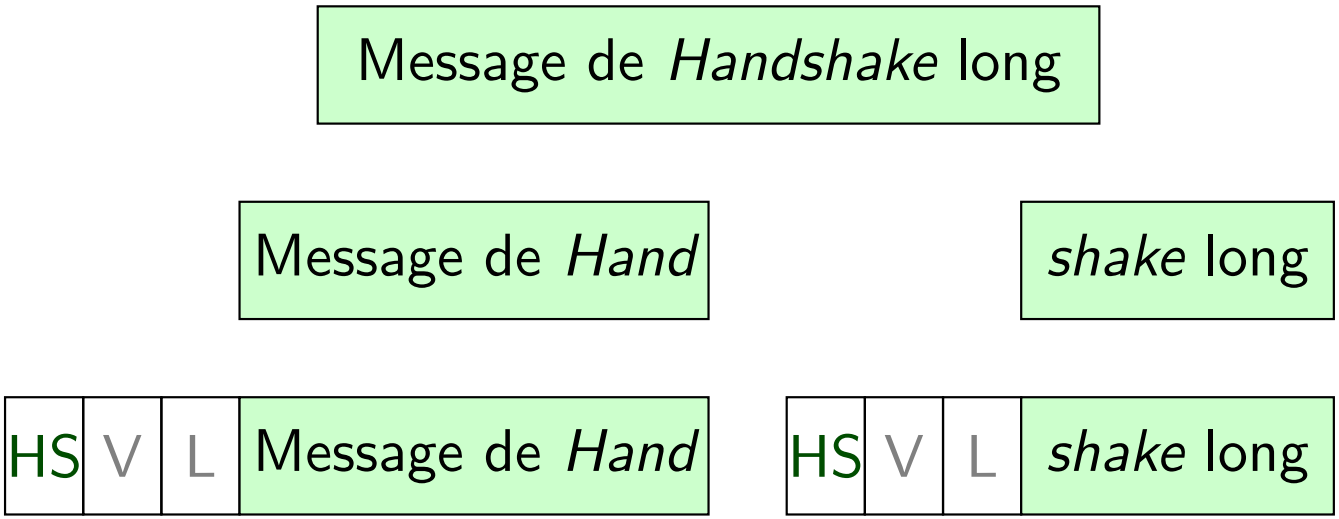
- divulgation d'information dans OpenSSL
- cause : lecture au-delà de la fin d'un tableau

Tous les messages sont découpés en *records*



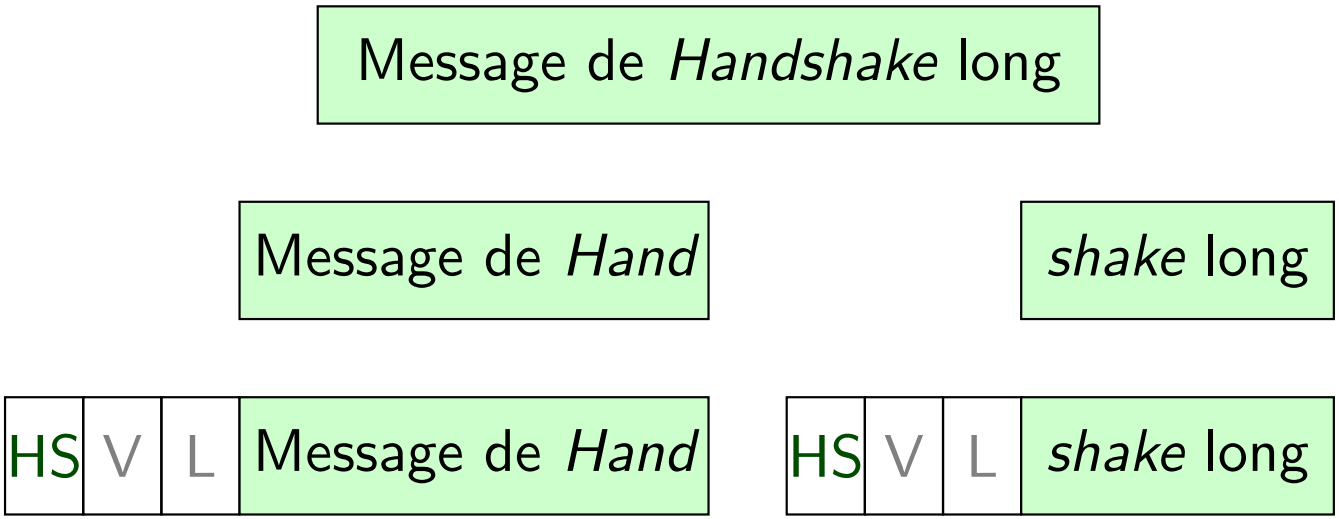
Retour sur *Heartbleed* : les *records*

Certains messages trop long doivent être envoyés sur plusieurs *records*

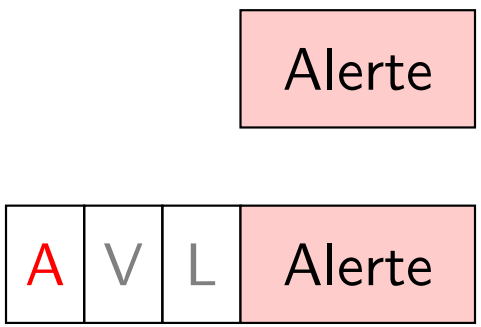


Retour sur *Heartbleed* : les *records*

Certains messages trop long doivent être envoyés sur plusieurs *records*

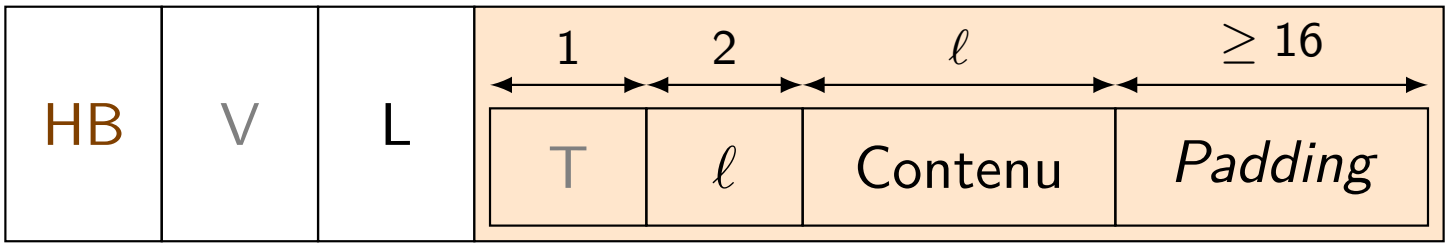


D'autres messages doivent être contenus dans exactement un *record*



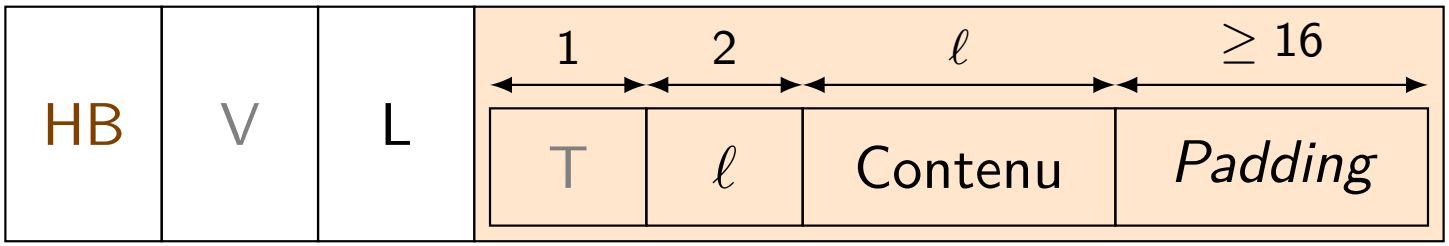
Retour sur *Heartbleed* : *Heartbeat* (RFC6520)

Les messages *Heartbeat* sont de taille variable, avec certaines contraintes imposées par la spécification



Retour sur *Heartbleed* : *Heartbeat* (RFC6520)

Les messages *Heartbeat* sont de taille variable, avec certaines contraintes imposées par la spécification

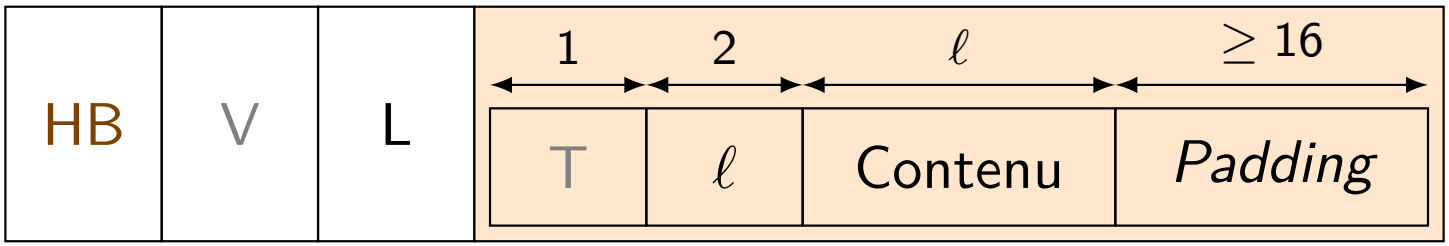


Que faire lorsque $L < \ell + 19$?

- rejeter le *record*

Retour sur *Heartbleed* : *Heartbeat* (RFC6520)

Les messages *Heartbeat* sont de taille variable, avec certaines contraintes imposées par la spécification

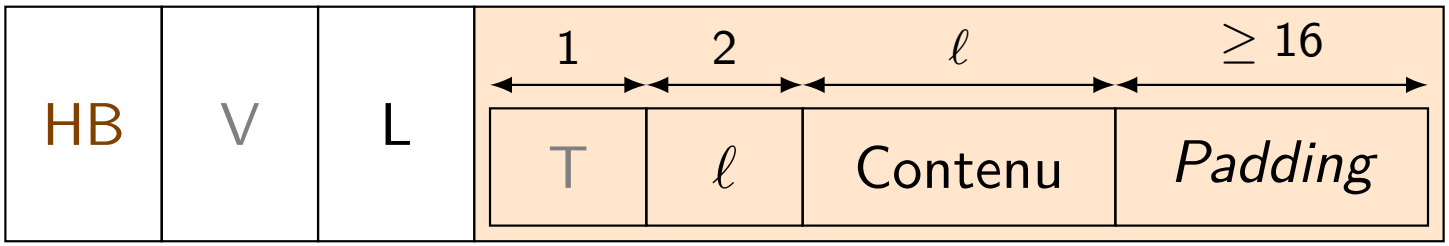


Que faire lorsque $L < \ell + 19$?

- rejeter le *record*
- attendre un *record* pour compléter

Retour sur *Heartbleed* : *Heartbeat* (RFC6520)

Les messages *Heartbeat* sont de taille variable, avec certaines contraintes imposées par la spécification

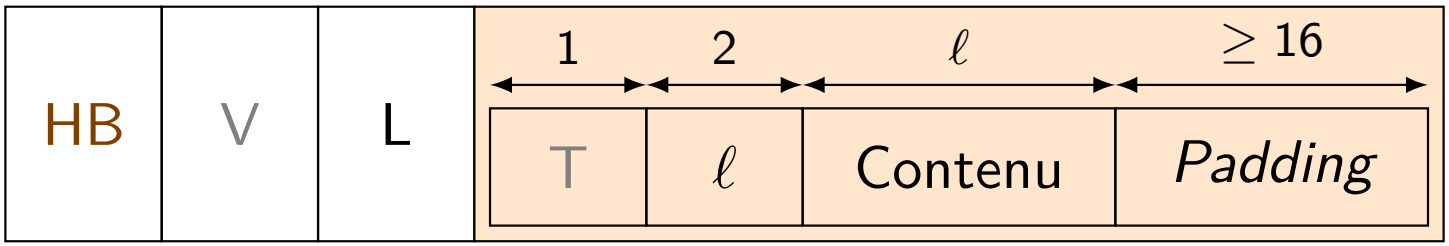


Que faire lorsque $L < \ell + 19$?

- rejeter le *record*
- attendre un *record* pour compléter
- **faire comme si tout allait bien et lire au-delà du tableau...**

Retour sur *Heartbleed* : *Heartbeat* (RFC6520)

Les messages *Heartbeat* sont de taille variable, avec certaines contraintes imposées par la spécification



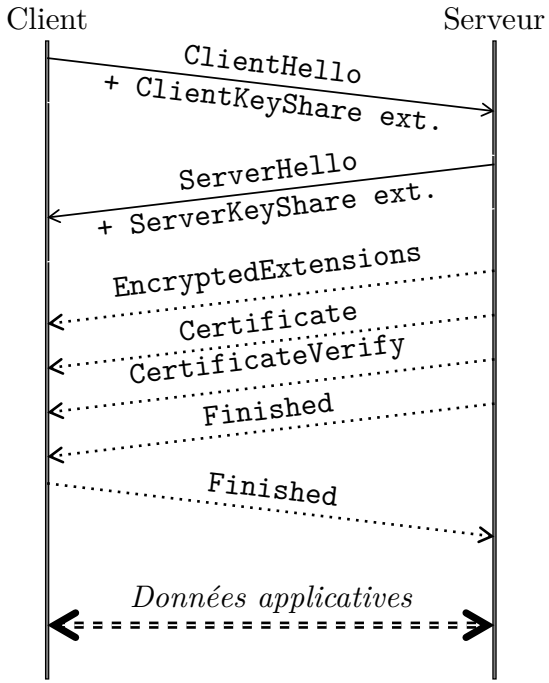
Que faire lorsque $L < \ell + 19$?

- rejeter le *record*
- attendre un *record* pour compléter
- **faire comme si tout allait bien et lire au-delà du tableau...**

La spécification devrait indiquer clairement qu'un *record* de type *Heartbeat* doit contenir **exactement un** message *Heartbeat*

Plan

Exemple d'une machine à état déficiente

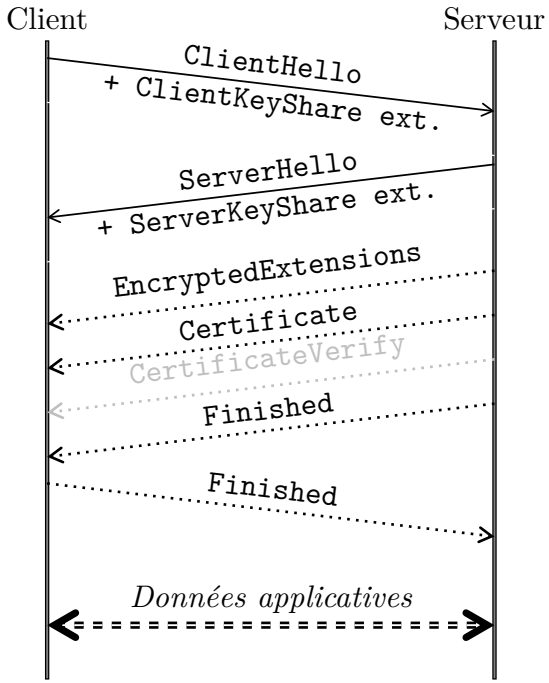


Dans TLS 1.3, en fonctionnement normal

- le serveur se présente (`Certificate`)
- il prouve son identité (`CertificateVerify`)
- ce message contient une signature qui requiert la clé privée du serveur

Travaux en cours avec A. Rasoamanana dans le cadre du projet GASP [RESSI20]

Exemple d'une machine à état déficiente



Dans TLS 1.3, en fonctionnement normal

- le serveur se présente (`Certificate`)
- il prouve son identité (`CertificateVerify`)
- ce message contient une signature qui requiert la clé privée du serveur

Que se passe-t-il si un client accepte une connexion où le `CertificateVerify` a été omis ?

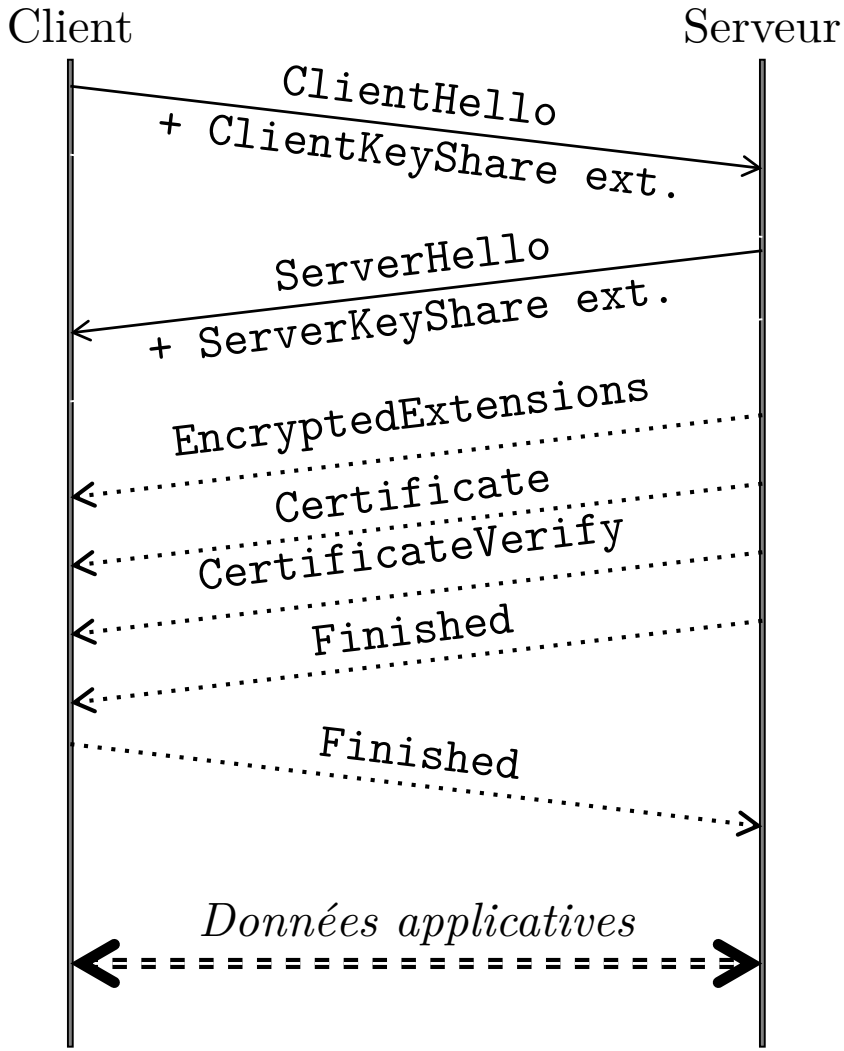
- il n'est plus nécessaire d'avoir la clé privée pour valider la négociation
- un attaquant peut usurper *n'importe quel serveur* vis-à-vis de ce client

Travaux en cours avec A. Rasoamanana dans le cadre du projet GASP [RESSI20]

Représentation de la machine à état du client

Représentation traditionnelle

- diagramme en « serpent »
- présentation du *happy path*



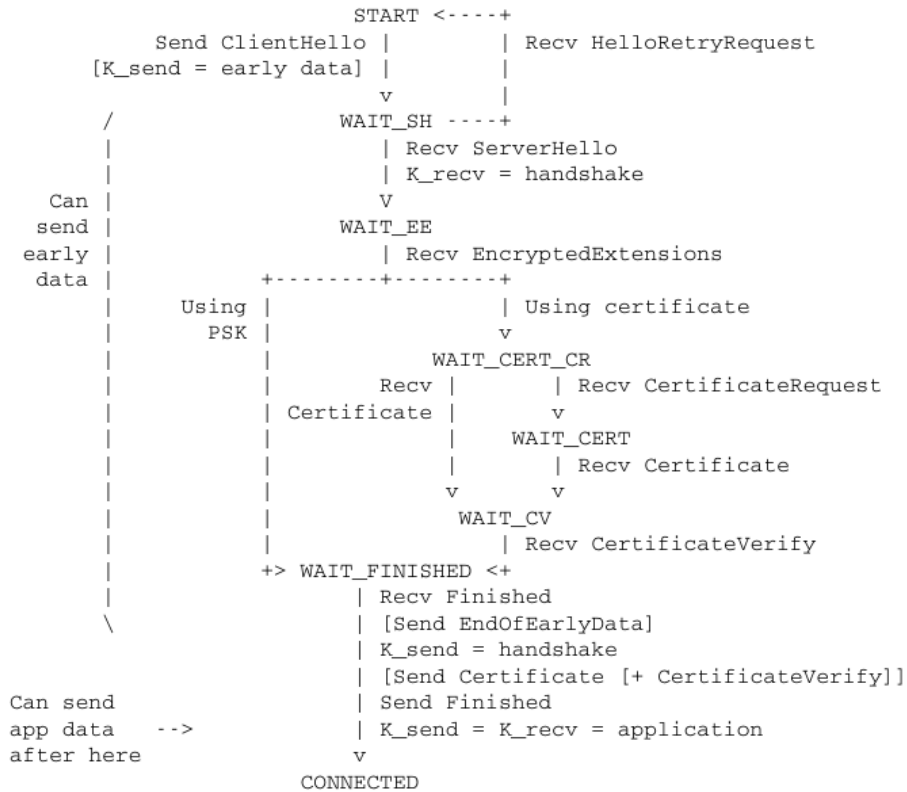
Représentation de la machine à état du client

Représentation traditionnelle

- diagramme en « serpent »
- présentation du *happy path*

Machine à états informelle

- effort de formalisation
- point de vue du client ici
- description encore ambiguë



— RFC 8446 (TLS 1.3) Annexe A

Représentation de la machine à état du client

Représentation traditionnelle

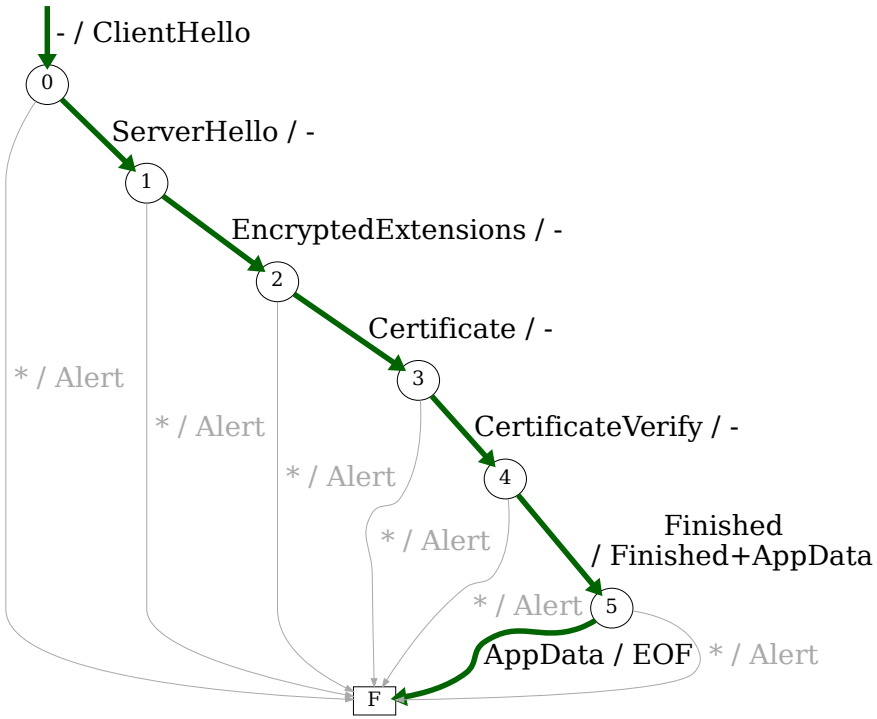
- diagramme en « serpent »
- présentation du *happy path*

Machine à états informelle

- effort de formalisation
- point de vue du client ici
- description encore ambiguë

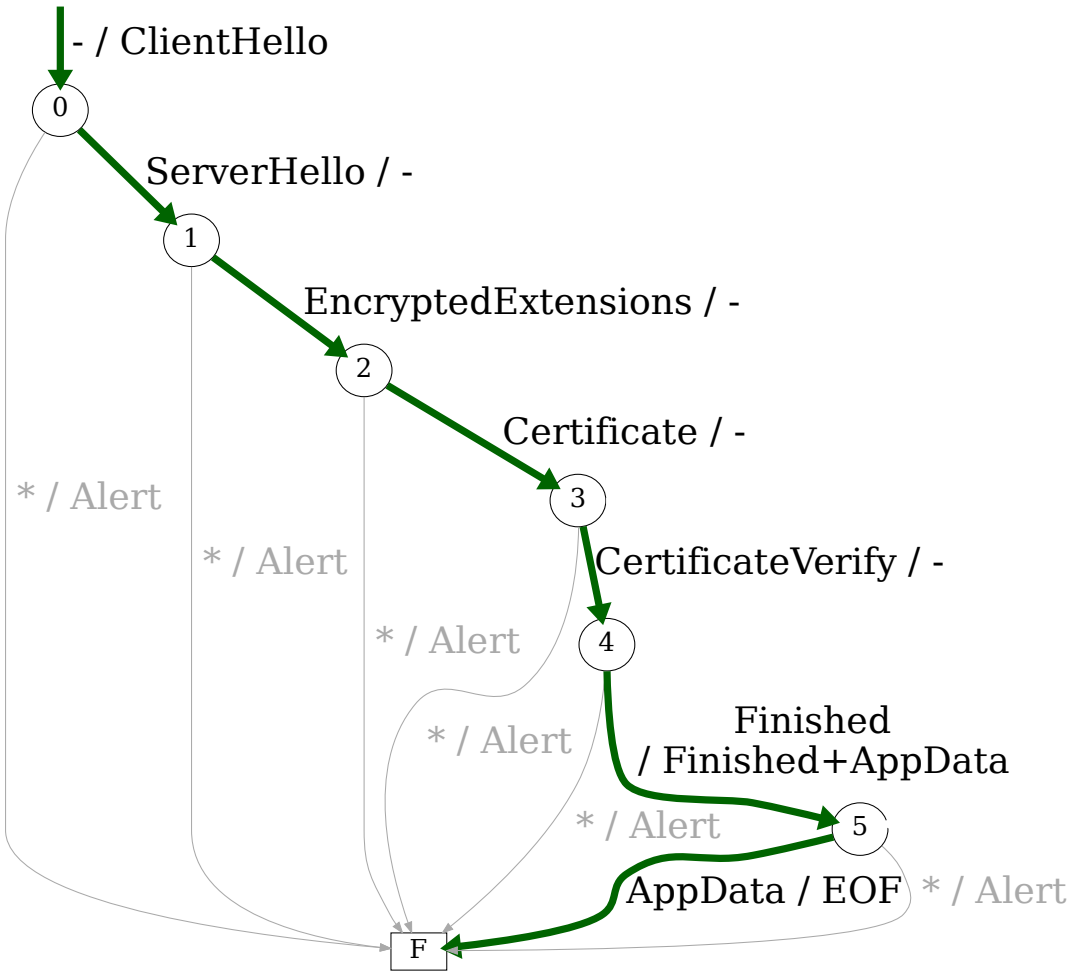
Machine de Mealy

- description formelle possible
- représentation plus lourde

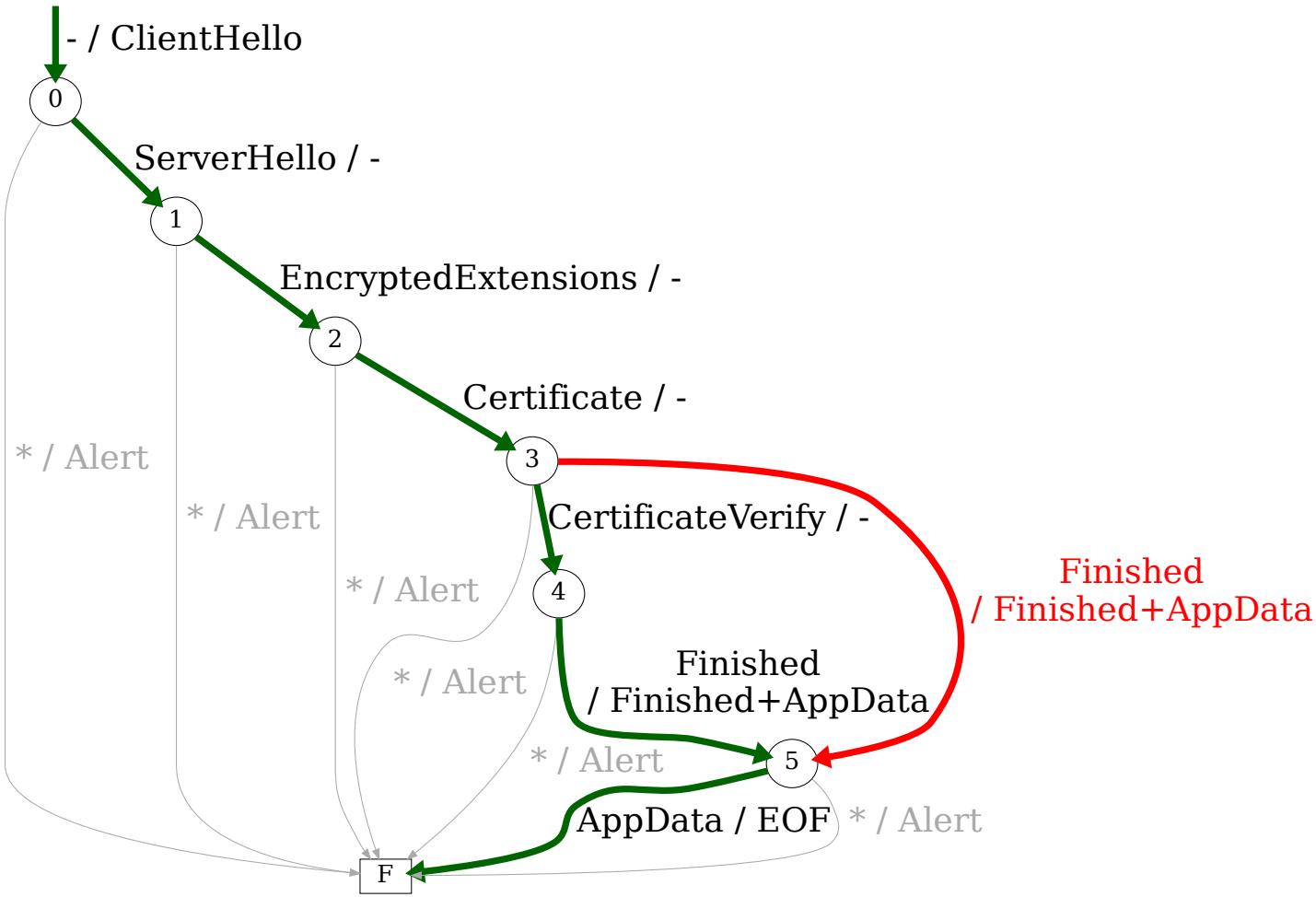


— Résultats du projet GASP

Mise en évidence de la CVE 2020-24613 sur wolfSSL



Mise en évidence de la CVE 2020-24613 sur wolfSSL



Inférence de machine à états

Possibilité d'inférer la machine à états d'une implémentation en boîte noire

- algorithme $L^\star$ décrit par Angluin en 1987
- adaptation aux machines de Mealy utilisée dans différents contextes
- inférence de machines à états de divers protocoles (ex. : TLS, H2)
- *(il existe d'autres approches, par exemple en altérant un transcript de référence par des mutations)*

Inférence de machine à états

Possibilité d'inférer la machine à états d'une implémentation en boîte noire

- algorithme $L^\star$ décrit par Angluin en 1987
- adaptation aux machines de Mealy utilisée dans différents contextes
- inférence de machines à états de divers protocoles (ex. : TLS, H2)
- *(il existe d'autres approches, par exemple en altérant un transcript de référence par des mutations)*

Application à TLS

- recherche de court-circuits de manière systématique
- recherche de boucles dans la machine à états
- utilisation des différences entre les machines à états inférées pour faire du *fingerprinting*

Un problème de spécification de la machine à états ?

Problème classique

- l'implémentation *réagit* aux messages reçus...
- ... sans restreindre l'ensemble des messages licites a priori
- nombreuses vulnérabilités en 2014 et 2015 sur TLS

Propositions

- utiliser une représentation formelle (machine de Mealy) comme spécification
- ne pas multiplier les chemins acceptables

Plan

Constats sur le développement de piles réseau

Les systèmes d'information reposent aujourd'hui sur des standards très complexes, ce qui a des conséquences variées sur la sécurité

- dénis de service
- exécution de code arbitraire
- confusion dans l'interprétation

Constats sur le développement de piles réseau

Les systèmes d'information reposent aujourd'hui sur des standards très complexes, ce qui a des conséquences variées sur la sécurité

- dénis de service
- exécution de code arbitraire
- confusion dans l'interprétation

Situation encouragée par plusieurs facteurs

- la tendance naturelle à vouloir que « ça marche »
- le refus de certains acteurs de contraindre les développeurs
- une partie de la communauté académique a tendance à considérer le problème de l'interprétation (*parsing*) comme un problème résolu
- certains langages de programmation ou outils sont fragiles

Utiliser des langages ou des outils modernes

Certaines classes de problèmes peuvent être détectées ou évitées

- gestion de la mémoire automatique
- prise en compte de la concurrence
- mise en œuvre d'un mode strict à la compilation
- intégration d'outils d'analyse lorsqu'ils existent

Il est également important de tester son code

- tester le négatif
- comparer le comportements de différentes implémentations
- éprouver la robustesse de son code (*fuzzing*)

Améliorer les spécifications

Problèmes principaux avec les spécifications actuelles

- des documents volumineux...
- ... rédigés en langage naturel

Éléments de solution : simplifier, formaliser, automatiser

Améliorer les spécifications

Problèmes principaux avec les spécifications actuelles

- des documents volumineux...
- ... rédigés en langage naturel

Éléments de solution : simplifier, formaliser, automatiser

Simplifier

- nouvel équilibre entre performance et simplicité
- penser à l'impact sur l'organisation du code de certains choix de conception

Améliorer les spécifications

Formaliser

- décrire les spécifications de manière précise
  - format des messages
  - transformations (découpage, multiplexage, crypto)
  - état interne
  - machine à états
- en utilisant des représentations classiques (BNF, automates)
- ou des DSL (ex. : [LangSec21])

Améliorer les spécifications

Formaliser

- décrire les spécifications de manière précise
  - format des messages
  - transformations (découpage, multiplexage, crypto)
  - état interne
  - machine à états
- en utilisant des représentations classiques (BNF, automates)
- ou des DSL (ex. : [LangSec21])

Automatiser

- une spécification formelle permet de tester certaines propriétés automatiquement...
- ... voire de dériver automatiquement des implémentations
- dans cet esprit, besoin de compilateurs

Améliorer les spécifications

Formaliser

- décrire les spécifications de manière précise
  - format des messages
  - transformations (découpage, multiplexage, crypto)
  - état interne
  - machine à états
- en utilisant des représentations classiques (BNF, automates)
- ou des DSL (ex. : [LangSec21])

Automatiser

- une spécification formelle permet de tester certaines propriétés automatiquement...
- ... voire de dériver automatiquement des implémentations
- dans cet esprit, besoin de compilateurs

Peut-on tout décrire et dériver à l'aide de représentations « simples » ?

Conclusion

- La complexité et l'ambiguïté sont deux ennemis de la sécurité
- De mauvais choix de conception peuvent mener à des implémentations fragiles
- Pistes pour améliorer les choses : simplifier, formaliser et automatiser
- À termes, peut-on envisager / souhaiter des implémentations purement descriptives ?

Questions ?

Merci pour votre attention

Références

[RESSI18] *Présentation d'un cours de programmation sécurisée*. OL. RESSI 2018

[LangSec16] *Caradoc : a pragmatic approach to PDF parsing and validation*. G. Endignoux, OL et J.-Y. Migeon. LangSec Workshop @ IEEE SSP 2016

[PhD16] *A study of the TLS ecosystem*. OL. Thèse soutenue en 2016

[CRiSIS20] *Implementations Flaws in TLS Stacks...*. OL, CRiSIS 2020

[RESSI20] *Le projet GASP : a Generic Approach to Secure network Protocols*. OL. RESSI 2020

[LangSec21] *Work-in-Progress : Towards a Platform to Compare Binary Parser Generators*. OL, S. Naud et A. T. Rasoamanana, LangSec Workshop @ IEEE SSP 2021

[ESORICS22] *Towards a Systematic and Automatic Use of State Machine Inference to Uncover Security Flaws and Fingerprint TLS Stacks*. A. T. Rasoamanana, H. Debar et OL, ESORICS 2022

Articles et ressources disponibles sur `https://paperstreet.picty.org` et `https://gasp.ebfe.fr`

Annexes

Illustration avec un code Mini-PNG invalide

```
assert f.read(8) == "Mini-PNG"

while not f.eof()
    block_type = f.read_char(1)
    block_length = f.read_int(4)
    block_content = f.read(block_length)

    if block_type == 'H'
        header = interpret_header(block_content)
        canvas = allocate(header.image_size)
        data_ptr = data
    elif block_type == 'D'
        copy(block_content, data_ptr)
        data_ptr += block_length
```

Retour sur *Heartbleed* : une meilleure spécification

Relativement à la situation précédente, la spécification indique

- *The total length of a HeartbeatMessage MUST NOT exceed* 2^{14} (la taille maximale d'un *record*)
- *If the payload_length of a received HeartbeatMessage is too large, the received HeartbeatMessage MUST be discarded silently.*

Retour sur *Heartbleed* : une meilleure spécification

Relativement à la situation précédente, la spécification indique

- *The total length of a HeartbeatMessage MUST NOT exceed* 2^{14} (la taille maximale d'un *record*)
- *If the payload_length of a received HeartbeatMessage is too large, the received HeartbeatMessage MUST be discarded silently.*

Ce n'est pas suffisant.
La spécification aurait *dû* indiquer :

- *A record with RecordType Heartbeat MUST contain exactly one HeartbeatMessage*
- *In particular, a HeartbeatMessage can not be split across multiple records*