Projet ANR-19-CE39-0001-01
a Generic Approach to Secure network Protocols

Olivier Levillain

Réunion de lancement des projets en sécurité globale
12 décembre 2019

Motivation

Quelques réalités sur les protocoles réseau

Motivation

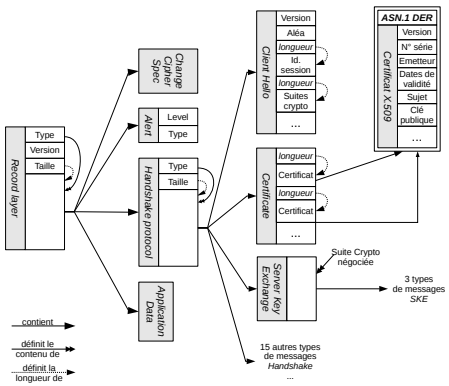
Quelques réalités sur les protocoles réseau

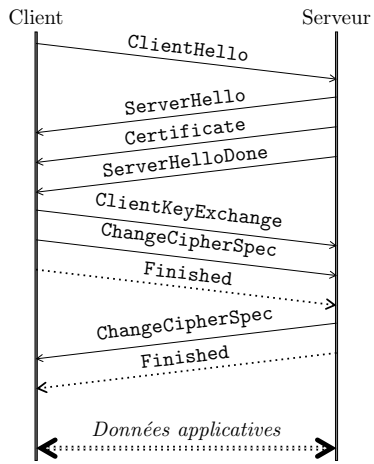
- ils sont partout

Motivation

Quelques réalités sur les protocoles réseau

- ils sont partout
- ils sont complexes

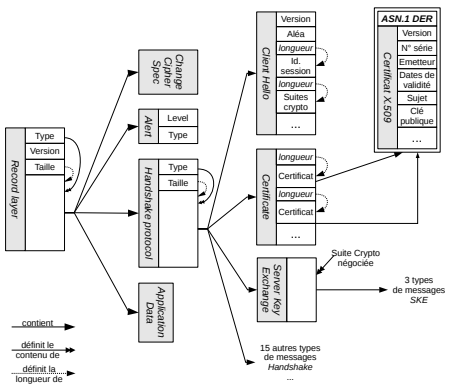


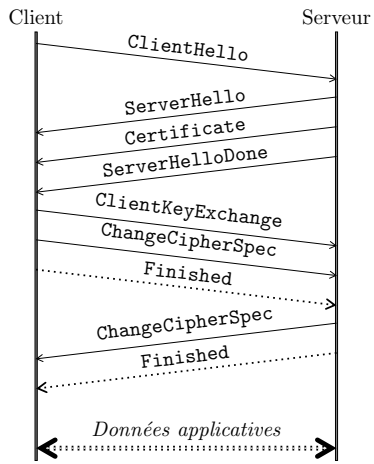


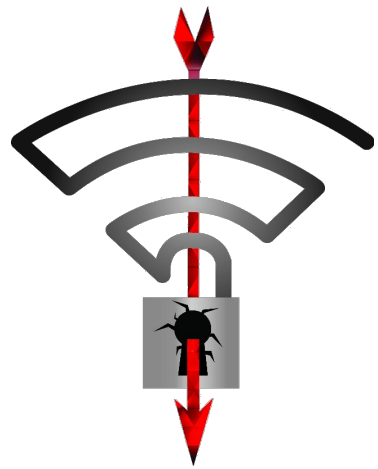
Motivation

Quelques réalités sur les protocoles réseau

- ils sont partout
- ils sont complexes
- leurs implémentations sont nombreuses et souvent vulnérables







Travaux existants

Analyse détaillée de certains protocoles, comme TLS

- implémentations robustes
- outils de tests, y compris au niveau protocolaire
- garanties formelles
- *ces travaux sont généralement peu transposables*

Travaux existants

Analyse détaillée de certains protocoles, comme TLS

- implémentations robustes
- outils de tests, y compris au niveau protocolaire
- garanties formelles
- *ces travaux sont généralement peu transposables*

Outils génériques pour tester les implémentations

- *fuzzers* de messages
- *fuzzers* protocolaires
- *ces outils manquent de précision (et parfois d'efficacité)*

GASP : *a Generic Approach to Secure Protocols*

Trois axes de travail

- *Network protocol observation in the field*
- *Protocol description to derive reference implementation*
- *Tests on existing implementations using a grey- or whitebox approach*

Méthodologie

- proposer des langages de description pour les messages et les machines à état
- dériver des implémentations de référence
- produire des outils de *scan* automatisé et des *fuzzers* à partir des descriptions

Description de messages : un exemple

Parsifal

- écriture de *parsers* (et de `dumpers`) grâce à du code **concis**
- **efficacité** des programmes produits
- **robustesse** des outils développés
- méthodologie de développement adaptée à l'écriture **incrémentale** de *parsers* flexibles

Description de messages : un exemple

Parsifal

- écriture de *parsers* (et de `dumpers`) grâce à du code **concis**
- **efficacité** des programmes produits
- **robustesse** des outils développés
- méthodologie de développement adaptée à l'écriture **incrémentale** de *parsers* flexibles

Limitations de Parsifal

- adhérence à OCaml...
- et en particulier à `camlp4`
- gestion baroque des formats non linéaires
- absence de garanties formelles

Description de messages : objectifs

Nouvelles idées

- regarder d'autres langages (Nom, Everparse, Recordflux, Nail, etc.)
- enrichir le DSL pour pouvoir raisonner sur les champs et messages
- meilleures gestion des contraintes
- meilleure séparation entre *parsing* et sémantique
- compilation vers différents langages (OCaml, Rust, $F^\star$, ADA/Spark)

Modélisation des machines à état

Animation de protocole

- un DSL pour décrire l'état interne d'un acteur
- un DSL pour décrire la machine à état à partir des messages, en incluant la manipulation de l'état
- un compilateur produisant des implémentations de référence
- un *fuzzer* de machines à état utilisant ces descriptions
- protocoles visés : TLS, BGP, H2, SSH, QUIC

Modélisation des machines à état

Animation de protocole

- un DSL pour décrire l'état interne d'un acteur
- un DSL pour décrire la machine à état à partir des messages, en incluant la manipulation de l'état
- un compilateur produisant des implémentations de référence
- un *fuzzer* de machines à état utilisant ces descriptions
- protocoles visés : TLS, BGP, H2, SSH, QUIC

Extension aux formats de fichiers

- l'interprétation des *chunks* PNG est décrit par une machine à état
- le traitement d'un fichier DVI

Utilisation de L* pour l'inférence d'automate (1/2)

Algorithme L*

- inférence d'automate à partir d'oracles (appartenance, équivalence)
- Dana Anglui — *Learning Regular Sets from Queries and Countermeasures*, 1987
- adaptation de l'algorithme avec des oracles réalistes

Utilisation de L* pour l'inférence d'automate (1/2)

Algorithme L*

- inférence d'automate à partir d'oracles (appartenance, équivalence)
- Dana Anglui — *Learning Regular Sets from Queries and Countermeasures*, 1987
- adaptation de l'algorithme avec des oracles réalistes

Interaction avec une implémentation en boîte noire

- concrétisation des messages à envoyer
- abstraction des messages reçus
- parcours de l'ensemble des branches possibles
- applications existantes à TLS, SSH et HTTP2

Utilisation de $L^\star$ pour l'inférence d'automate (2/2)

Modélisation de l'oracle d'appartenance par l'analyse du comportement du serveur

- réponse par un message
- erreur
- *time out*

Utilisation de L* pour l'inférence d'automate (2/2)

Modélisation de l'oracle d'appartenance par l'analyse du comportement du serveur

- réponse par un message
- erreur
- *time out*

Amélioration envisagées

- détection des *timeouts* par introspection de l'outil analysé
- ajout d'un mécanisme de gel / *fork* / reprise pour accélérer l'exploration
- extension des alphabets à des messages plus détaillés ou à des messages corrompus

Questions ?

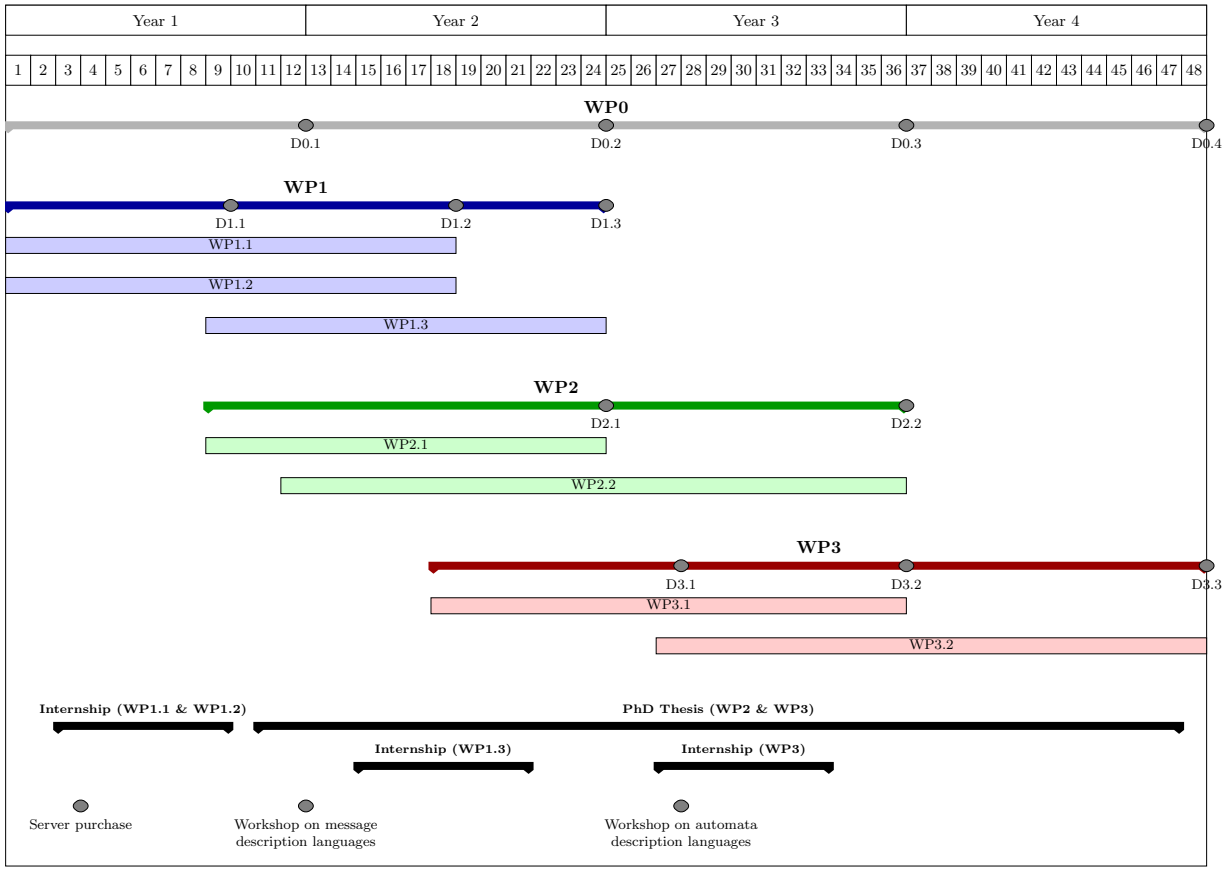
Merci de votre attention

Annexes

Livrables et tâches (1/2)

WP0	Project management and dissemination	
	D0.*	Yearly progress reports
WP1	Network protocol observation in the field	
	WP1.1	Specification of a message description language
	WP1.2	Development of compilers to derive parsers
	WP1.3	Measurement campaigns
	D1.1	Intermediate report on the message language and compilers
	D1.2	Final report on the message language and compilers
	D1.3	Campaigns results (tools, data and analyses)
WP2	Protocol description to derive reference implementations	
	WP2.1	Specification of a protocol description languages
	WP2.2	Development of compilers to derive reference implementations
	D2.1	Intermediate report on the languages and compilers
	D2.2	Final report on the languages and compilers
WP3	Tests on existing implementations using a grey- or whitebox approach	
	WP3.1	Test tools derived from the description languages
	WP3.2	Program introspection to explore implementation behaviour
	D3.1	Intermediate report on test tools
	D3.2	Final report on test tools
	D3.3	Report on implementation introspection

Livrables et tâches (2/2)



Idées pour le nouveau DSL (P2)

Exemple sur le format de fichier PNG :

```
struct png_chunk = {
  chunk_size : uint32;
  chunk_type : string(4);
  chunk_data : chunk_content;
  chunk_crc : uint32;
} constraints {
  chunk_size = len(chunk_content);
  chunk_crc = crc32(chunk_type ^ chunk_data);
  chunk_type = discriminant (chunk_content)
}
```