State Machine Issues in Network Stacks and Application to TLS and SSH

Aina Rasoamanana

Séminaire des Étudiants et Anciens
Télécom SudParis
2025-10-23

Plan

Plan

Context: Specification and Implementations

- Network protocols are defined in specs such as RFCs
- They are written in English (and not in a formal language)
  - ambiguities
  - incomplete specifications

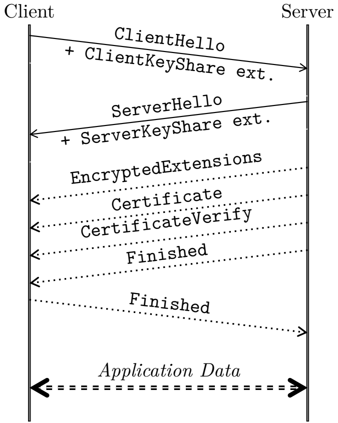
Context: Specification and Implementations

- Network protocols are defined in specs such as RFCs
- They are written in English (and not in a formal language)
  - ambiguities
  - incomplete specifications

In this presentation, we focus on state machine issues

- e.g. CVE-2020-24613
- (server authentication bypass in TLS)

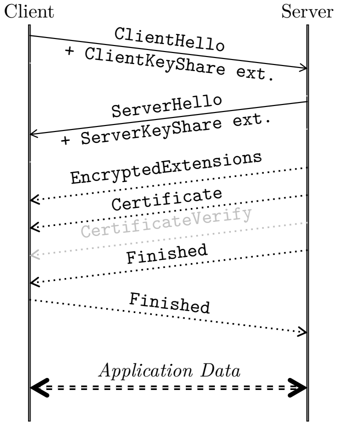
CVE-2020-24613: The Flaw



In a normal TLS 1.3 message flaw

- the server presents its (`Certificate`)
- it proves its identity (`CertificateVerify`)
- this message contains a signature (requiring the private key)

CVE-2020-24613: The Flaw

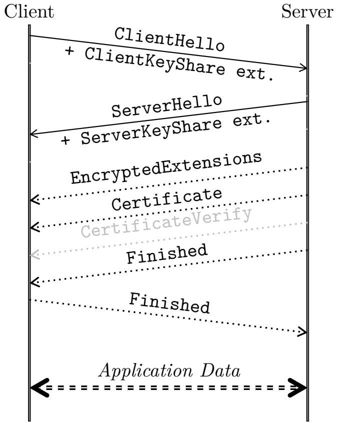


In a normal TLS 1.3 message flaw

- the server presents its (`Certificate`)
- it proves its identity (`CertificateVerify`)
- this message contains a signature (requiring the private key)

What happens if a client accepts a connection missing the `CertificateVerify`?

CVE-2020-24613: The Flaw



In a normal TLS 1.3 message flaw

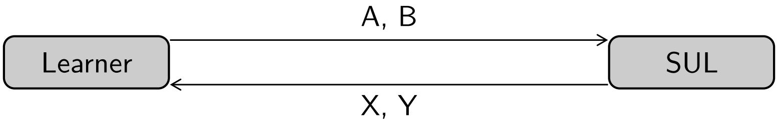
- the server presents its (`Certificate`)
- it proves its identity (`CertificateVerify`)
- this message contains a signature (requiring the private key)

What happens if a client accepts a connection missing the `CertificateVerify`?

- the private key is not necessary anymore for a successful handshake
- an attacker can impersonate *any server* to such a client

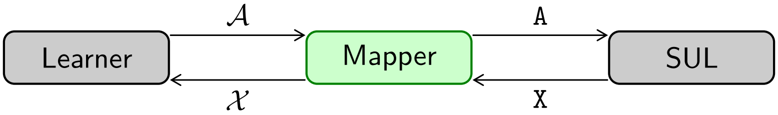
Plan

Methodology: Pipeline



SUL = System Under Learning (the stack being analyzed)

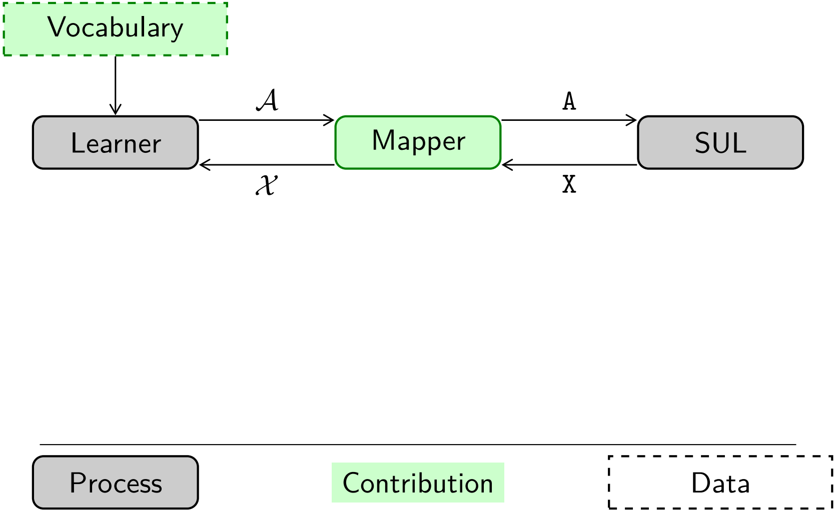
Methodology: Pipeline



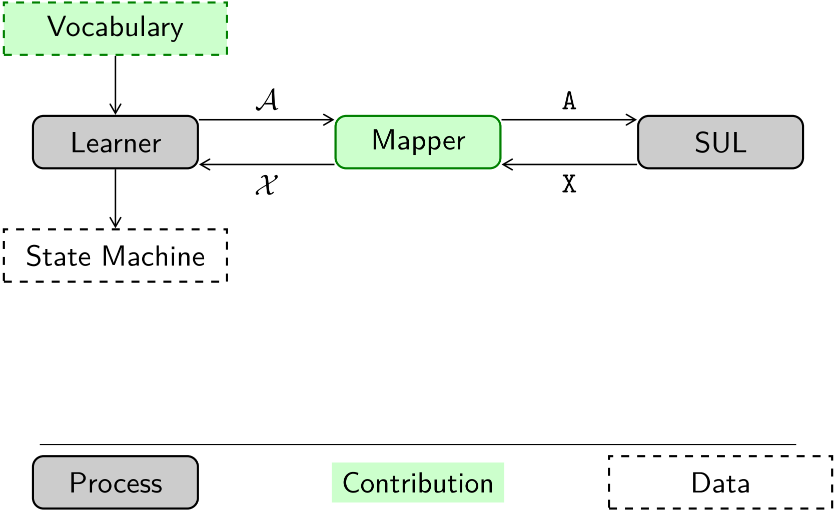
Real interactions require an intermediate component

- sending messages one at a time
- using concrete messages (bytes on the wire)

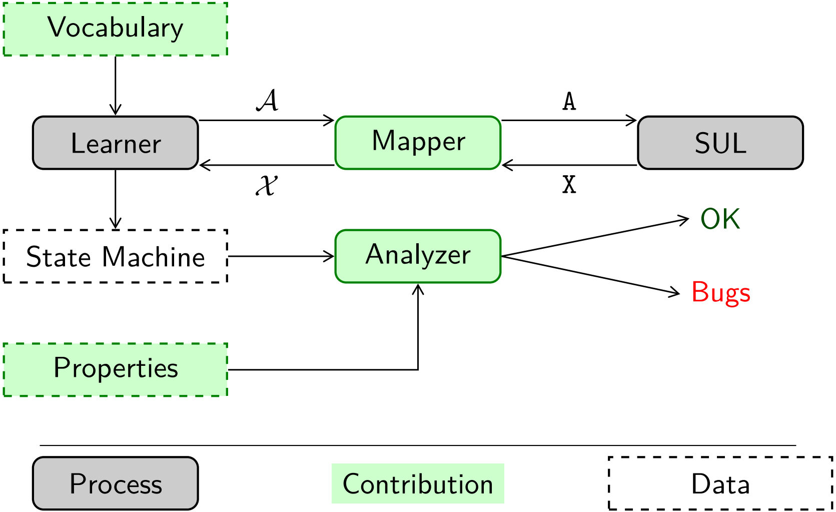
Methodology: Pipeline



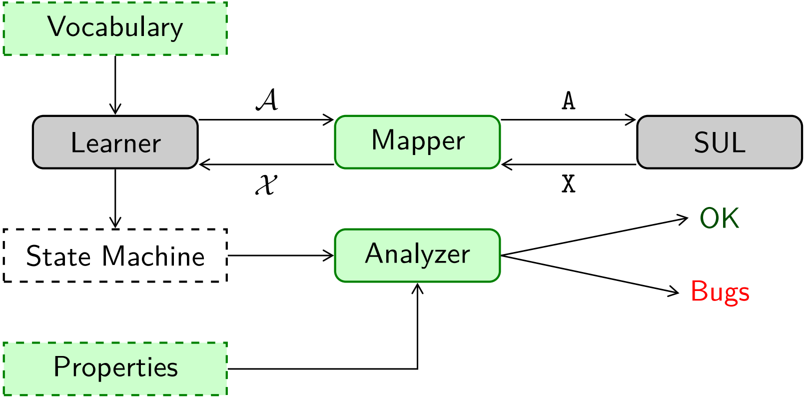
Methodology: Pipeline



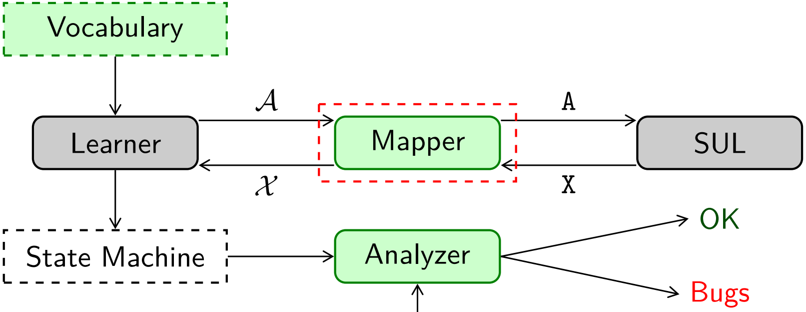
Methodology: Pipeline



Methodology: Pain Points



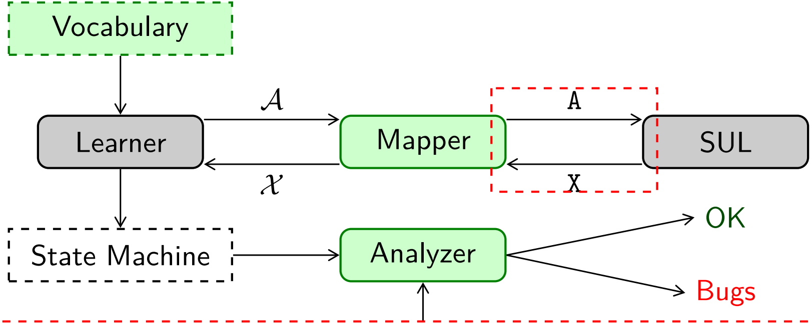
Methodology: Pain Points



Writing a new mapper for each protocol

- we (almost) need a protocol implementation...
- which is flexible and robust
- where every choice is ideally clear and explicit

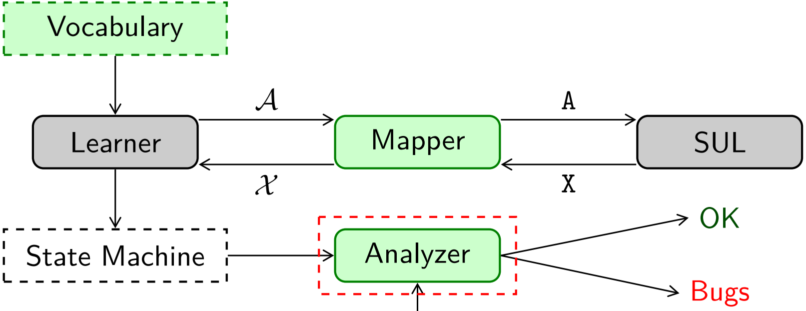
Methodology: Pain Points



Handling timeouts

- we do not know when SUL is done talking
- low values lead to missed messages
- high values lead to inefficiency

Methodology: Pain Points



Analyzing the state machines

- the resulting automata can be huge and hard to read
- need for automatic tools for adhoc properties

Plan

Studied Protocols

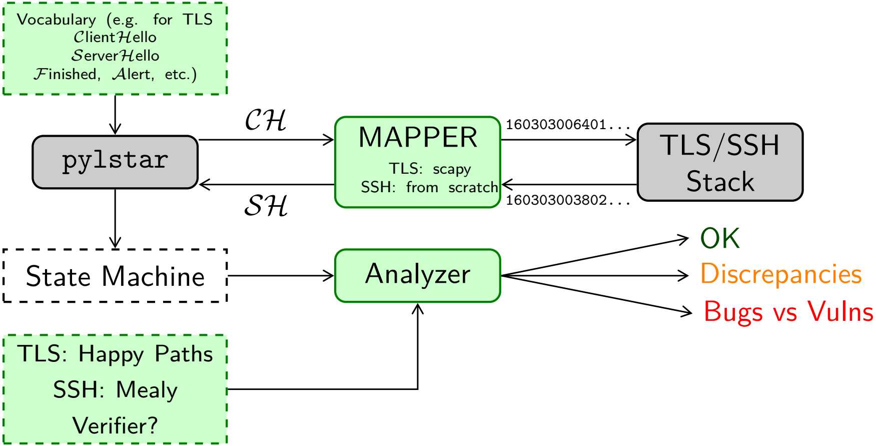
TLS

- the **S** of HTTP**S**
- one of the fundamental block of internet security

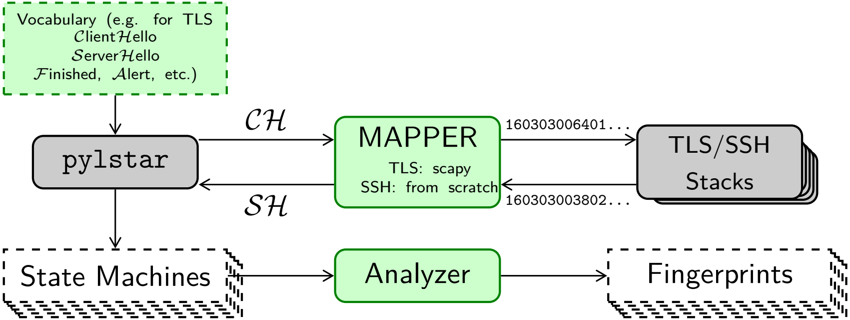
SSH

- remote secure shell and file copy
- designed to replace older cleartext protocols

Bug and Vulnerability Detection



Fingerprinting



TLS and SSH stacks actually vary in their behavior

- the distinguishing sequences lead to fingerprints
- possibly more robust than other techniques

SSH in a Nutshell (1/2)

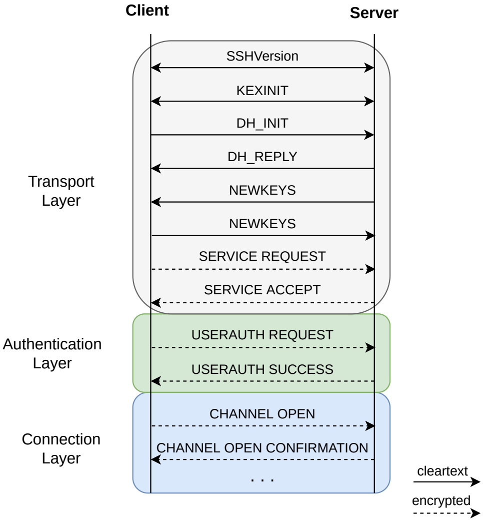
SSH in a Nutshell (1/2)

- SSH means SSH-2 in this presentation

SSH in a Nutshell (1/2)

- SSH means SSH-2 in this presentation
- SSH is a 3-layer protocol
  - Transport: key exchange, server authentication and channel protection
  - User Authentication
  - Connection: multiplexing application data channels

SSH in a Nutshell (2/2)



Challenges with SSH State Machine Inference

SSH is an *interesting* protocol

Challenges with SSH State Machine Inference

SSH is an *interesting* protocol

- more messages than TLS (20-30)
  - Generic (`Disconnect`, `ServiceRequest`, `Unimplemented`...)
  - Transport (`KexInit`, `(EC)DHInit`, `NewKeys`...)
  - UserAuthentication (`AuthRequest`, `AuthSuccess`, `AuthFailure`...)
  - Connection (`ChannelOpen`, `ChannelData`, `ChannelEOF`...)

Challenges with SSH State Machine Inference

SSH is an *interesting* protocol

- more messages than TLS (20-30)
- more states
  - the Transport layer is similar to TLS
  - the UserAuthentication layer should be rather simple
  - the Connection layer complexity depends on the modeling
  - after the initial handshake, keys can be refreshed

Challenges with SSH State Machine Inference

SSH is an *interesting* protocol

- more messages than TLS (20-30)
- more states
- non-deterministic behavior
  - some stacks implement defensive timeouts during the first layers
  - the order of some messages can vary from one run to the other

Challenges with SSH State Machine Inference

SSH is an *interesting* protocol

- more messages than TLS (20-30)
- more states
- non-deterministic behavior
- non-representable behavior
  - opening channels in the midst of a key refresh operation
  - how to represent server-side timeouts in a useful way

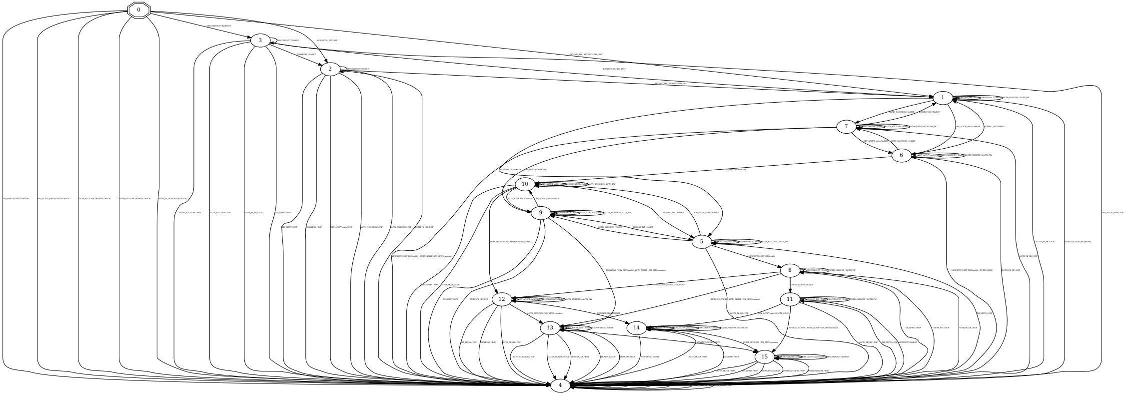
Challenges with SSH State Machine Inference

SSH is an *interesting* protocol

- more messages than TLS (20-30)
- more states
- non-deterministic behavior
- non-representable behavior

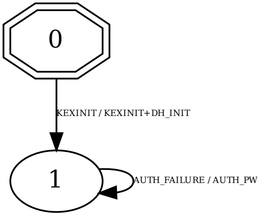
A typical TLS inference was in the minutes...
With SSH it can take hours or days

Case Study: mysteressh



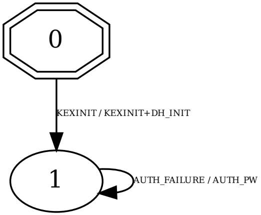
- Client state machine
- Transport + UserAuthentication layers
- 8 input messages, 16 states

Early `UserAuthFailure` (`password` Flavor)



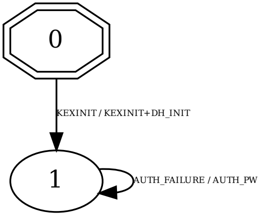
- If a rogue server starts the handshake...
- and sends an early `UserAuthFailure` with the `password` method...

Early `UserAuthFailure` (`password` Flavor)



- If a rogue server starts the handshake...
- and sends an early `UserAuthFailure` with the `password` method...
- the vulnerable client happily sends its password (`AUTH_PW`)

Early UserAuthFailure (password Flavor)



- If a rogue server starts the handshake...
- and sends an early `UserAuthFailure` with the `password` method...
- the vulnerable client happily sends its password (`AUTH_PW`)
- Problem: there was no server authentication

Plan

Ongoing and future work

Short-term, on SSH

- a paper on the challenges and our answers
- discussion with editors about strange behavior

Ongoing and future work

Short-term, on SSH

- a paper on the challenges and our answers
- discussion with editors about strange behavior

Long-term

- more efficient inferences (adaptive learning, system-level tricks)
- more efficient analyses/visualisation

Ongoing and future work

Short-term, on SSH

- a paper on the challenges and our answers
- discussion with editors about strange behavior

Long-term

- more efficient inferences (adaptive learning, system-level tricks)
- more efficient analyses/visualisation

Longer-term

- towards automatic mappers
- extension to other domains

Questions

Thank you for your attention

References

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

[ARES24] *Mealy Verifier: An Automated, Exhaustive, and Explainable Methodology for Analyzing State Machines in Protocol Implementations*. A. Tran Van, OL et H. Debar, ARES 2024