Fast identification of network protocol states with greybox active automata learning
(Ongoing work)

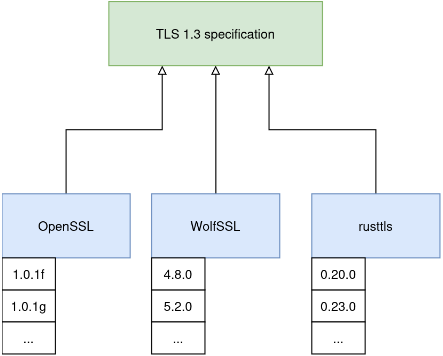
PIPEREAU Yohan   MICHEL Mathieu   LEVILLAIN Olivier

Télécom SudParis

September 30, 2024

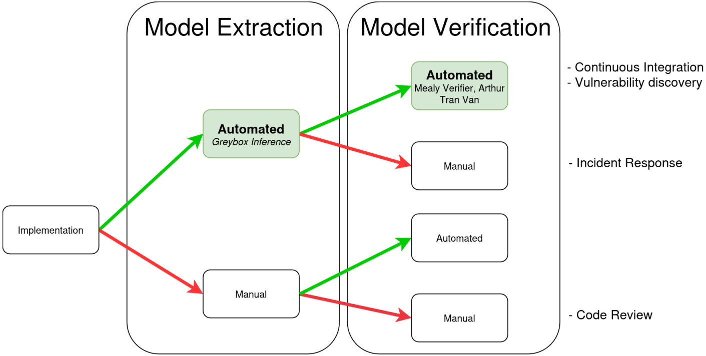
Introduction to protocol inference

Goal: Identify protocol deviation (security)



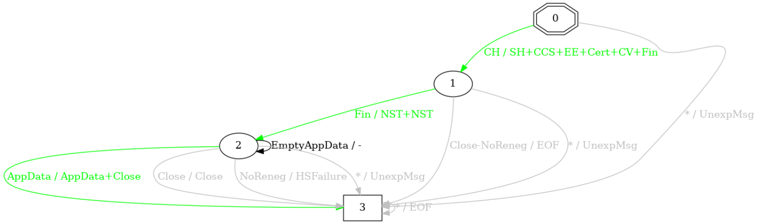
Multiple implementations and versions for a same standard

Automating protocol verification



Global picture

Model for implementation: Mealy Machines



OpenSSL 3.0.13 TLSv1.3 generated using pylstar-tls

Active Automata learning: MAT Framework

Minimally Adequate Teacher (MAT) answers

- Membership queries
- Equivalence queries

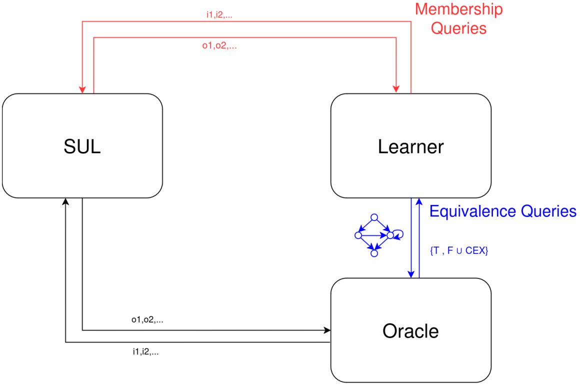


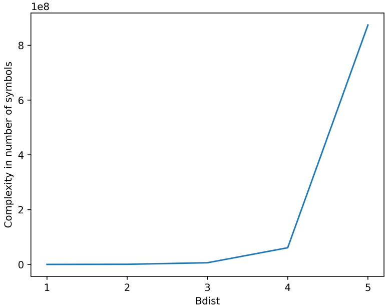
Figure: MAT framework c.f. Arthur Tran Van

Most of AAL time is spent in equivalence queries

Equivalence method: exhaustiveness vs duration

Guess parameter k: number of states, or length of separating sequence, ...

- k too low → Miss states
- k too high → Loose time ($O(|I|^k)$)



Symbol complexity of L* with BDist EqMethod for BitVise state machine (MS3=7)

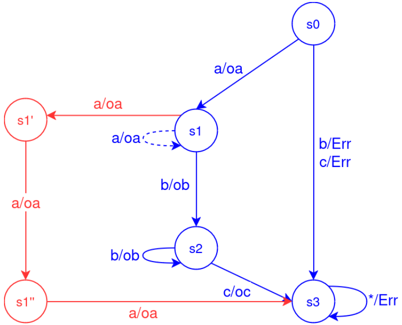
1

[1] An Experimental Evaluation of Conformance Testing Techniques in Active Automata Learning, Bharat Garhewal et al., MODELS 2023

How to improve the average complexity of equivalence queries ?

Principles of I/O equivalence method

- Input: An hypothesis Mealy machine, upper-bound on lookahead
- Goal: Finding a counter-example
  1. Verification => State, Transition coverage
  2. Discovery => Lookahead
- **Conformance Testing** [2]: W, WP, HSI, ...



Equivalence Method
Analogy of Noreneg in WolfSSL 4.8.0 for TLSv1.2

[2] Conformance Testing, Angelo Gargantini, LNCS 3472, 2005

Intuition

Memory for state separation

- Same memory $\rightarrow$ Same state
- Different memory $\rightarrow$ Maybe different states

Complexity to be exhaustive

- I/O : ∞
- Memory: $O(V + E)$

Oracle based on debug object graph classification

Object Graph properties

Definition - Object (Di)Graph $G(V, E)$

- V: An object
  - Virtual address
  - Value
  - Type
- E: A reference or a embedding
  - Relationship: Contains, PointsTo
  - Name

Object Graph Properties

- **roots**
- **leaves**
- may have **cycles**: dynamic datastructures : e.g. double linked lists
- often **disconnected**: disjoint subgraph starting from each root

Why debug object graph ?

	Memory Snapshots	**Object Graphs**	**Debug Object Graphs**
Object ID	Virtual address	Graph Path	Graph Path
Object Nature	Live & Dead	Live	Live
Object values	✓	✓	✓
Memory Topology	✗	✓	✓
Object Types	✗	✗	✓
Object Names	✗	✗	✓

How to build a Debug Object Graph ?

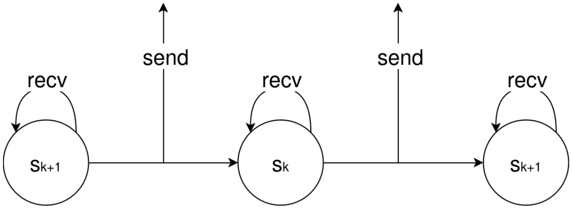
Roots

- CPU Registers
- Stack Frames variables
- Global variables

Dwarf Type Tree

- **Container Types**: Enum, Structure, Union
- **Primitive Types**: Basic Types, Array
- **Type Aliases**: Const Type, Typedef

When to snapshot ?



Snapshot Point

Snapshot point

On each FSM state, the server is either:

- waiting for the creation of a new connection: accept()
- waiting for a message: recv(), read(), . . .
- crashed

Our contribution: DwarfGC

DwarfGC: Build an Object Graph with debug symbol

- Shared library
- C lang
- 3244 LOC

	# Vertex	# Edges
OpenSSL 1.0.1g	[5282; 6387]	[5947; 7097]
WolfSSL 4.8.0	[343; 374]	[397; 432]

Number of Vertex and relationship

Can we infer the same machine as in blackbox AAL ? without debug symbols ?

$G_{s1} = G_{s2}$? $G_{s1} \approx G_{s2}$!

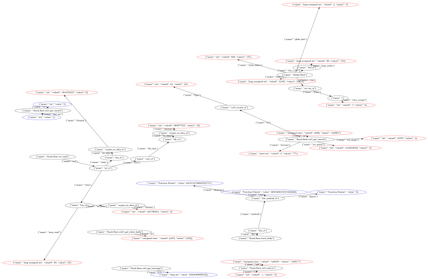
	Comparison	Difficulty
Topological comparison	isomorphism $G_{s1} \cong G_{s2}$	Multiple connections
Value comparison	Value equality $V_{s1} = V_{s2}$	Random objects, Uninitialized objects

Method 1: Graph Kernels and Classification

- $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$
- Problem: Common denominator object graph

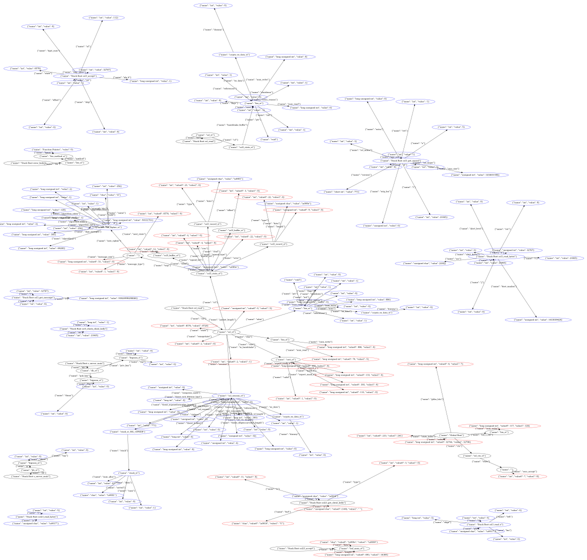
Method 2: Learning Graph transformations (Ongoing)

- Associate (State, Input Symbol) with a set of graph transformations
- Hidden Markov Chain ?



Same states

- TLS12ClientHelloRSA
- TLS12ClientHelloRSA, TLSApplicationDataEmpty



Different logical state
- TLS12ClientHelloRSA, TLSApplicationDataEmpty, TLSApplicationDataEmpty
- TLSApplicationDataEmpty

Conclusion

Goal

- **Step 1** Can we infer the same machine as in blackbox AAL ? without debug symbols ?
- **Step 2** Can we infer states invisible in blacbox AAL ?

Contribution

DwarfGC: an object graph tracer using debug symbols

Future work

$G_{s1} \approx G_{s2}$

Advertisement

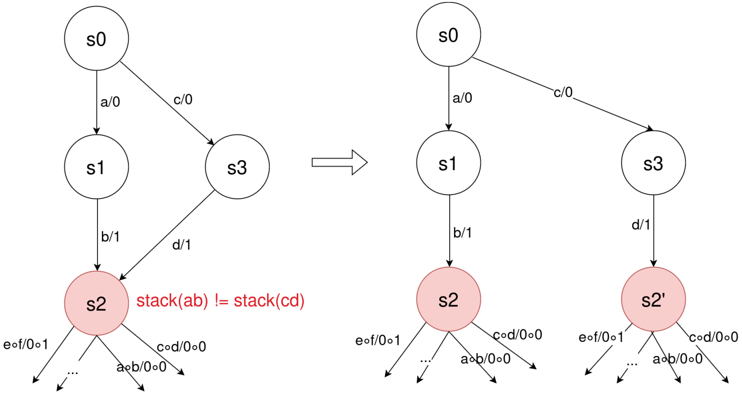
https://github.com/stanp-org/

Appendix

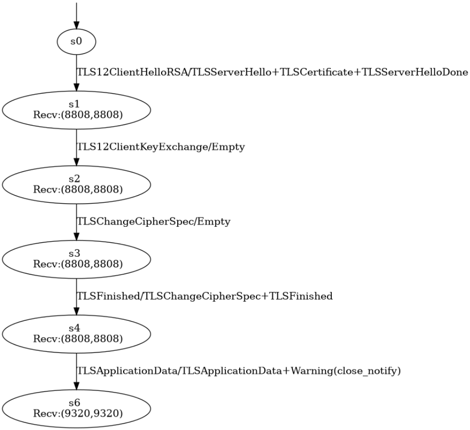
Idea - Greybox Equivalence

Stack size of $a \bullet b$ and $c \bullet d$ are different

1. **Longer sep-sequence**: $\exists k > bdist, \exists w \in I^k, (c \bullet d) \bullet w \neq (a \bullet b) \bullet w$
2. **Unknown isym**: $\exists i \notin I, \exists prefix \in I^k, (c \bullet d) \bullet (prefix \bullet i) \neq (a \bullet b) \bullet (prefix \bullet i)$
3. **Unknown osym**: $o \in O, \exists (o1, o2) \notin O, o \to (o1, o2)$
4. **Internal state**: CFG node uncaptured by input and output

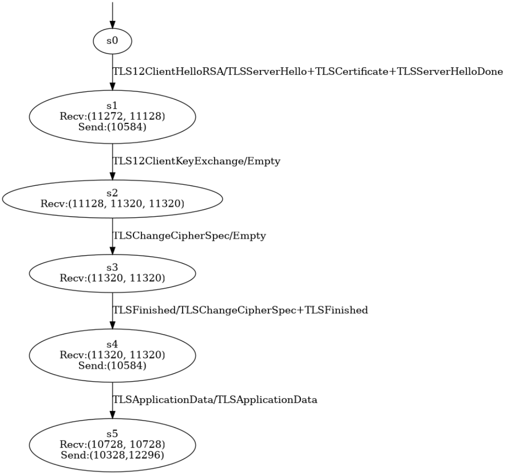


WolfSSL Happy Path with callstack oracle



WolfSSL v4.8.0 TLSv1.2 with stack size on states

OpenSSL Happy Path with callstack oracle

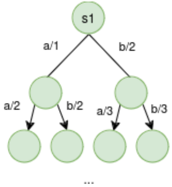


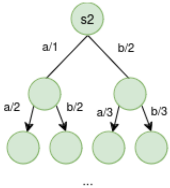
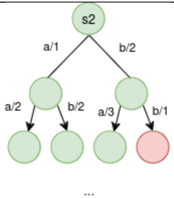
OpenSSL 1.0.1g TLSv1.2 with stack size on states

Key properties used to implement I/O Eq Method

Definition

Kleene Closure of I $I^* = \epsilon, a, b, aa, bb, ab, ba, \ldots$



s_1 I/O equivalent s_2	s_1 I/O separated from s_2
$s_1 \equiv s_2$	$w \sim s_1 \# s_2$
$\forall w \in I^*, Output(s_1, w) = Output(s_2, w)$	$\exists w \in I^*, Output(s_1, w) \neq Output(s_2, w)$
	

Memory interface

Definition - Memory equivalent states

Two states $s_1, s_2 \in S$ are **memory equivalent** iff for each state-defining variables of the program the value in the memory snapshot at state s_1 is equal to the value in the memory snapshot at state s_2

Definition - state-defining object $Smem(s_1)$

State-defining objects are the set of objects which uniquely characterize a FSM state.

Building an accurate oracle for equivalence query <=> Finding the complete list of objects which can characterize a state uniquely