Finding Security Vulnerabilities in Protocol Implementations Using Active Automata Learning

Frits Vaandrager

Radboud University Nijmegen

ICGI, Wroclaw, September 2018
Research Question

SUT

What's going on inside this black box?

input 1

input 2

output 1

output 2
Minimally adequate teacher (Angluin)

- Learner
  - input sequences
  - MQ
  - output sequences
  - hypothesis
  - EQ
  - counterexample

- Teacher

Frits Vaandrager
Finding Bugs Using Automata Learning
Black box checking (Peled, Vardi & Yannakakis)

Learner: Formulate hypotheses
Conformance Tester (CT): Test correctness hypotheses
Introduction
Case Studies
Register Automata
Taint Analysis
Conclusions and Future Work

Black box checking (Peled, Vardi & Yannakakis)

Learner: Formulate hypotheses
Conformance Tester (CT): Test correctness hypotheses

Model learning and conformance testing two sides of same coin!
Welcome to the LearnLib home page! LearnLib is a free, open-source (Apache License 2.0) Java library for active automata learning. It is mainly being developed at the Chair for Programming Systems at TU Dortmund University, Germany; a complete list of contributors can be found on the team page.

**Note:** The open-source LearnLib is a from-scratch re-implementation of the former closed-source version. See the features page for a comparison of the feature sets of the two versions.

**Background**

- Read some Papers on LearnLib
- Papers citing LearnLib at Google Scholar

**EXTERNAL LINKS**

- LearnLib @ GitHub
- AutomataLib @ GitHub

**RECENT POSTS**

- Open Source release of LearnLib
Research method (FMSD, 2015)
A theory of mappers

Learner

abstract input

small $\sum$ (equivalence classes)

abstract output

Mapper

concrete input

Teacher

concrete output

probably large $\sum$

Abstract model
A theory of mappers (cnt)

Formally, a mapper can be viewed as a transducer (deterministic Mealy machine). A mapper $\mathcal{A}$ induces an abstraction operation $\alpha_{\mathcal{A}}$ and a concretization operator $\gamma_{\mathcal{A}}$.

**Theorem**

For a mapper $\mathcal{A}$ and nondeterministic Mealy machines $\mathcal{M}$ and $\mathcal{H}$, $\alpha_{\mathcal{A}}(\mathcal{M}) \leq \mathcal{H}$ implies $\mathcal{M} \leq \gamma_{\mathcal{A}}(\mathcal{H})$

**Theorem**

Suppose $\alpha_{\mathcal{A}}(\mathcal{M})$ has no transitions with output $\bot$. Then $\mathcal{M} \leq \gamma_{\mathcal{A}}(\mathcal{H})$ implies $\alpha_{\mathcal{A}}(\mathcal{M}) \leq \mathcal{H}$.
EMV protocol (Aarts et al, 2013)

- EMV = Europay/Mastercard/Visa
- Compatibility between smartcards and terminals
- SEPA requires EMV compliance
- EMV standard has >700 pages
- Learning took at most 1500 membership queries, less than 30 minutes
- Useful for fingerprinting cards
E.dentifier2 (WOOT’14)

Finding Bugs Using Automata Learning

Introduction
Case Studies
Register Automata
Taint Analysis
Conclusions and Future Work

E-bankieren ABN Amro kwetsbaar

donderdag 16 aug 2012, 10:02 (Update: 17-08-12, 08:39)

Internetbankieren bij ABN Amro is gevoelig voor fraudes. Internetkriminalen kunnen sommige betalingstransacties onderschappen, aanpassen en doorsluizen naar hun eigen rekening.

Frits Vaandrager
State machines for old and new E.dentifier2
Bugs in protocol implementations

Standard violations found in implementations of major protocols:

- **TLS** (Usenix Security’15)
- **TCP** (CAV’16)
- **SSH** (Spin’17).
Bugs in protocol implementations

Standard violations found in implementations of major protocols:

- **TLS** (Usenix Security’15)
- **TCP** (CAV’16)
- **SSH** (Spin’17).

These findings led to bug fixes in implementations.
Learned model for SSH implementation
### SSH model checking results

<table>
<thead>
<tr>
<th>Property</th>
<th>Key word</th>
<th>OpenSSH</th>
<th>Bitvise</th>
<th>DropBear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>Trans.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Auth.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rekey</td>
<td>Pre-auth.</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Auth.</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Funct.</td>
<td>Prop. 6</td>
<td>MUST</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Prop. 7</td>
<td>MUST</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Prop. 8</td>
<td>MUST</td>
<td>X*</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Prop. 9</td>
<td>MUST</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Prop. 10</td>
<td>MUST</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Prop. 11</td>
<td>SHOULD</td>
<td>X*</td>
<td>X*</td>
</tr>
<tr>
<td></td>
<td>Prop. 12</td>
<td>MUST</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Other case studies

- Session Initiation Protocol (SIP)
- Message Queuing Telemetry Transport (MQTT) protocol
- Quick UDP Internet Connections (QUIC) protocol
- WiFi
- IEC 60870-5-104 protocol
- ...
Lorentz Workshop

Participants from automata learning, model-based testing, cryptography, and security protocol implementation.

Working groups on e.g.,
- WiFi
- side channels in TLS
- LTE
Welcome to the Automata Wiki!

Active automata learning is emerging as a highly effective technique for obtaining models of protocol implementations and other reactive systems. Many algorithms have been proposed in the literature. Often variations of these algorithms exist for different classes of models, e.g., DFAs, Moore machines, Mealy machines, interface automata, and various forms of register automata. Algorithms for generation of conformance test suites often play a crucial role within active automata learning, as an oracle to determine whether a learned model is correct or not, and also here we see a wide variety of algorithms that have been proposed for different model classes.

Although there has been some excellent experimental work on evaluating algorithms for learning and conformance testing, the number of realistic models used for benchmarking is rather limited, and different papers use different industrial cases. Often the benchmarks used are either small/academic, which do not properly evaluate efficiency, or randomly generated, and it is clear from the experiments that performance of algorithms on randomly generated models is often radically different from performance on models of real systems that occur in practice.

A mature field is characterized by the presence of a rich set of shared benchmarks that can be used to compare different approaches. We have therefore set up this wiki with a publicly available set of benchmarks of state machines that model real protocols and embedded systems. These benchmarks will allow researchers to compare the performance of learning and testing algorithms.

We invite all our colleagues to contribute and send us (links to) other benchmarks they know of, for inclusion in the wiki.
Overview benchmarks

The benchmark models are stored per benchmark and grouped within a benchmark per model type. Below we first list the benchmarks per model type and finally we list all benchmarks combined. For more details about how the benchmark models are stored within this wiki we refer to the GlobalDataStructure page in the Help section.

The wiki also support tagging of benchmarks, e.g. the page Tag/Protocol lists all benchmarks tagged as protocol. See the page Tag for all defined tags.

In the Table page we list the statistics, such as number of states, for the register and mealy benchmarks.

Per model type

DFA benchmarks

- BenchmarkRandomDFAs
- BenchmarkToyDFA

Moore machine benchmarks

- BenchmarkRandomMoore
- BenchmarkToyMoore

Mealy machine benchmarks

- BenchmarkBankcard
- BenchmarkCircuits
- BenchmarkCoffeeMachine
- BenchmarkIdentifier2
- BenchmarkESMcontroller
- BenchmarkFromRhapsodyToDecyze
- BenchmarkIECG0870-5-104
Automata wiki: supported automata frameworks

- LTS
  - genMoore
  - genMealy
  - IA
  - DFA
  - Moore
  - Mealy
  - DFIA
  - Register
  - FSM
### Automata wiki: most benchmarks are not that big

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Size</th>
<th>Error</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>BenchmarkTLS/GnuTLS_3.3.12_client_regular.dot</td>
<td>7</td>
<td>8/10</td>
<td>56</td>
</tr>
<tr>
<td>BenchmarkTLS/GnuTLS_3.3.8_server_regular.dot</td>
<td>12</td>
<td>8/10</td>
<td>96</td>
</tr>
<tr>
<td>BenchmarkTLS/GnuTLS_3.3.12_server_full.dot</td>
<td>9</td>
<td>12/12</td>
<td>108</td>
</tr>
<tr>
<td>BenchmarkTLS/RSA_BSAFE_C_4.0.4_server_regular.dot</td>
<td>9</td>
<td>8/11</td>
<td>72</td>
</tr>
<tr>
<td>BenchmarkTCP/TCP_Windows8_Server.dot</td>
<td>38</td>
<td>13/10</td>
<td>494</td>
</tr>
<tr>
<td>BenchmarkTCP/TCP_FreeBSD_Server.dot</td>
<td>55</td>
<td>13/11</td>
<td>715</td>
</tr>
<tr>
<td>BenchmarkTCP/TCP_Linux_Server.dot</td>
<td>57</td>
<td>12/9</td>
<td>684</td>
</tr>
<tr>
<td>BenchmarkTCP/TCP_Linux_Client.dot</td>
<td>15</td>
<td>10/11</td>
<td>150</td>
</tr>
<tr>
<td>BenchmarkTCP/TCP_Windows8_Client.dot</td>
<td>13</td>
<td>10/12</td>
<td>130</td>
</tr>
<tr>
<td>BenchmarkTCP/TCP_FreeBSD_Client.dot</td>
<td>12</td>
<td>10/12</td>
<td>120</td>
</tr>
<tr>
<td>BenchmarkX-ray-system-PCS/learnresul5.dot</td>
<td>9</td>
<td>12/9</td>
<td>66</td>
</tr>
<tr>
<td>BenchmarkX-ray-system-PCS/learnresul4.dot</td>
<td>7</td>
<td>9/7</td>
<td>37</td>
</tr>
<tr>
<td>BenchmarkX-ray-system-PCS/learnresul6.dot</td>
<td>9</td>
<td>12/9</td>
<td>66</td>
</tr>
<tr>
<td>BenchmarkX-ray-system-PCS/learnresul3.dot</td>
<td>7</td>
<td>9/7</td>
<td>37</td>
</tr>
<tr>
<td>BenchmarkX-ray-system-PCS/learnresul2.dot</td>
<td>3</td>
<td>9/4</td>
<td>9</td>
</tr>
<tr>
<td>BenchmarkX-ray-system-PCS/learnresul1.dot</td>
<td>8</td>
<td>9/7</td>
<td>44</td>
</tr>
<tr>
<td>BenchmarkESMcontroller/run3_hyp.101.obf.dot</td>
<td>3410</td>
<td>78/151</td>
<td>265980</td>
</tr>
<tr>
<td>BenchmarkESMcontroller/run1_hyp.135.obf.dot</td>
<td>3410</td>
<td>78/151</td>
<td>265980</td>
</tr>
</tbody>
</table>
Actions may carry data parameters that may be stored in registers:
Register automata may be parametrized by a (relational) structure: a pair $\langle \mathcal{D}, \mathcal{R} \rangle$ where $\mathcal{D}$ is an unbounded domain of data values, and $\mathcal{R}$ is a collection of relations on $\mathcal{D}$.

Examples of simple structures include:

- $\langle \mathbb{N}, \{=\} \rangle$, the natural numbers with equality;
- $\langle \mathbb{R}, \{<\} \rangle$, the real numbers with inequality: this structure also allows one to express equality between elements.

Transition guards are conjunctions of negated and unnegated relations from $\mathcal{R}$.
Learning tools for register automata

- **Tomte**, Radboud University, can only handle $\langle \mathbb{N}, \{=\} \rangle$
- **LearnLib**, TU Dortmund, can only handle $\langle \mathbb{N}, \{=\} \rangle$
- **RALib**, Uppsala/Dortmund, can handle some richer structures
TCP protocol case study (FMICS-AVoCS’17)

These findings led to bug fix in Linux TCP implementation.

Frits Vaandrager

Finding Bugs Using Automata Learning
These findings led to bug fix in Linux TCP implementation.
Limits of black-box learning?

- Model learning is an highly effective bug finding technique
Limits of black-box learning?

- Model learning is an highly effective bug finding technique
- ... but it has some serious scalability problems
Limits of black-box learning?

- Model learning is an highly effective bug finding technique
- ... but it has some serious scalability problems
- Can we use white-box information while preserving the extensionality of black-box models?
Idea: Use Taint Analysis
Taint Analysis

- White-box technique for code analysis
- Instruments code to track input values
- Many tools focus on specific vulnerabilities, e.g. buffer overflows and sql injections
- Usually implemented using Dynamic Binary Analysis, e.g. Valgrind
- We use Python library from Pygmalion tool from Andreas Zeller et al.
What Does Pygmalion Tool Do For Us?

Potential of exponential gains during learning!
What Does Pygmalion Tool Do For Us?

Potential of exponential gains during learning!
Architecture RAlib Tool for Learning Register Automata

- **SL\* Algorithm**
  - tree query
  - SDTs

- **Tree Oracle**
  - tests
  - observations

- **SUL**
Tree Oracle

Diagram showing a tree structure with nodes labeled with logical expressions and transitions labeled with functions like `OOut(z)` and `true`.
Replace tree oracle in RAlib by a version that uses taint analysis.
Replace tree oracle in RAlib by a version that uses taint analysis.

First prototype finished (for integers with equality)!!!
Active automata learning is emerging as a highly effective bug-finding technique, and slowly becoming a standard tool in the toolbox of the software engineer.

But much further research is needed!
Future Work

1. Explore combinations of black-box and white-box learning
2. Further improvement of learning and testing algorithms for FSM models
3. Develop algorithms for models with time and probabilities
4. Refactoring of legacy software is potentially excellent application domain
5. Please contribute benchmarks to our repository!