Back In Black:
Towards Formal, Black Box Analysis Of Sanitizers and Filters

George Argyros*, Ioannis Stais**, Angelos Keromytis* and Aggelos Kiayias***
Motivation

• Sanitizers and filters are important components of securing applications.
  - Think code injection attacks.
• Black-Box analysis is often a necessity.
  - Penetration testing, hardware testing.
• Filters need to be fast.
  - Possibility of representing with automata models.
• This talk: focus on regular expression filters.
  - Check the paper for results on sanitizers.
Regular Expression Filters

• Pass untrusted input through Regular Expressions.
  - Reject if match found.

• Widely employed for protecting against code injection attacks.
  - Not very robust.

• Significant components of large scale software.
  - Web Application Firewalls, IDS, DPI and others.

• Represented by Deterministic Finite State Automata (DFA).
Can we efficiently infer Regular Expression Filters?
Exact Learning From Queries

Form of Active Learning.

Two types of Queries.
Exact Learning From Queries

Membership Query

Is $s$ accepted by $M$?
Exact Learning From Queries

**Equivalence Query**

Is $M = H$? Yes, or provide counterexample.
Learning Deterministic Finite Automata

[Angluin ’87], [Rivest-Schapire ’93]

- Start with an initial state.
- Test all transitions from that state.
- When valid DFA is formed test for Equivalence.
- Counterexamples provide access to previously undiscovered states.

Testing all transitions is inefficient for large Alphabets!
Symbolic Finite Automata (SFA)

Classical Automata

Symbolic Automata

guards
Learning SFA: Challenges

• Alphabet may be infinite!

• How to distinguish causes for counterexamples in the models?
  - Counterexamples due to undiscovered states in the target.
  - Counterexamples due to inaccurate transition guards.
Learning Symbolic Finite Automata

- Start with an initial state.
- Test **sample** transitions from that state.
- Use sample transitions as training set to generate guards.
- Novel counterexample processing method to handle incorrect guards.

Convergence under natural assumptions on guardgen()
Is Exact Learning From Queries a realistic model?
Is Exact Learning from Queries a realistic model?

- Membership Queries? *Test whether input is rejected by the filter.*
- Equivalence Queries?
Grammar Oriented Filter Auditing

or

How to Implement an Equivalence Oracle
Grammar Oriented Filter Auditing (GOFA)
Grammar Oriented Filter Auditing (GOFA)

Context Free Grammar $G$

select_exp: SELECT name
any_all_some: ANY | ALL
column_ref: name
parameter: name
Grammar Oriented Filter Auditing (GOFA)

Context Free Grammar \( G \)

\[
\text{select_exp: SELECT name}
\]
\[
\text{any_all_some: ANY | ALL}
\]
\[
\text{column_ref: name}
\]
\[
\text{parameter: name}
\]
Grammar Oriented Filter Auditing (GOFA)

Context Free Grammar $G$

select_exp: SELECT name
any_all_some: ANY | ALL
column_ref: name
parameter: name

Regular Filter $F$

(/index.php?id=1' or '1'=‘1

Normal output or REJECT

{alter{s}"+[w]+.*character{s}+set{s}+[w]+)\";{s}+waitfor{s}+time{s}+)
Grammar Oriented Filter Auditing (GOFA)

May Require Exponential Number of Queries!

Normal output or REJECT

/index.php?id=1' or '1'='1

Context Free Grammar $G$

Regular Filter $F$

\{\text{alter|w|+.*character|s|+set|w|+(\\;|s|*\text{waitfor|s|+time|s|+\|})}\}
Solving GOFA

• In an ideal (White-Box) world both $G$ and $F$ are available:
  1. Compute $\bar{F}$, the set of strings not rejected by $F$.
  2. Check $\mathcal{L}(G \cap \bar{F})$ for emptiness.

• In practice $F$ is unavailable.
  - Learn a model for $F$!
Solving GOFA

Context Free Grammar $G$

Regular Filter $F$
Solving GOFA

Context Free Grammar $G$

Regular Filter $F$
Solving GOFA

*Membership Query*

Context Free Grammar $G$

Regular Filter $F$

string $s$

$True$ if REJECT is returned

$False$ otherwise
Solving GOFA

**Equivalence Query**

One Membership Query per Equivalence Query!

If no such \( s \) exists then terminate.

If \( \text{REJECT} \):
\( s \) is a counterexample for \( H \).

Otherwise:
\( s \) is a bypass for the filter \( F \).
Evaluation
Experimental Setup

- 15 Regular Expression Filters from popular Web Application Firewalls (WAFs).
  - 7 - 179 states.
  - 13 - 658 transitions.
- Alphabet size of 92 symbols.
  - Includes most printable ASCII characters.
DFA vs SFA Learning

- On average 15x less queries.
- Increase in Equivalence queries.
- Speedup is not a simple function of the automaton size.
# DFA vs SFA Learning

<table>
<thead>
<tr>
<th>IDS RULES</th>
<th>DFA LEARNING</th>
<th>SFA LEARNING</th>
<th>SPEEDUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>STATES</td>
<td>ARCS</td>
<td>MEMBER</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>13</td>
<td>4389</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>35</td>
<td>21720</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>33</td>
<td>56834</td>
</tr>
<tr>
<td>4</td>
<td>33</td>
<td>38</td>
<td>102169</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>155</td>
<td>193109</td>
</tr>
<tr>
<td>6</td>
<td>60</td>
<td>113</td>
<td>250014</td>
</tr>
<tr>
<td>7</td>
<td>66</td>
<td>82</td>
<td>378654</td>
</tr>
<tr>
<td>8</td>
<td>70</td>
<td>99</td>
<td>445949</td>
</tr>
<tr>
<td>9</td>
<td>86</td>
<td>123</td>
<td>665282</td>
</tr>
<tr>
<td>10</td>
<td>115</td>
<td>175</td>
<td>1150938</td>
</tr>
<tr>
<td>11</td>
<td>135</td>
<td>339</td>
<td>1077315</td>
</tr>
<tr>
<td>12</td>
<td>139</td>
<td>964</td>
<td>1670331</td>
</tr>
<tr>
<td>13</td>
<td>146</td>
<td>380</td>
<td>1539764</td>
</tr>
<tr>
<td>14</td>
<td>164</td>
<td>191</td>
<td>2417741</td>
</tr>
<tr>
<td>15</td>
<td>179</td>
<td>658</td>
<td>770237</td>
</tr>
</tbody>
</table>

**AVG** = 15.31
GOFA Algorithm Evaluation

• Assume that the grammar G does not contain a string that bypasses the filter.
  - How good is the approximation of the filter obtained?
  - How efficient is SFA Learning in the GOFA context?
• What is an appropriate grammar to perform this experiment?
  - Use the filter itself as the input grammar!
  - Intuitively, a maximal set that does not include a bypass.
DFA vs SFA Learning in GOFA

✓ SFA utilizes x35 less queries.

✓ States recovered:
  ‣ DFA: 91.95%
  ‣ SFA: 89.87%
GOFA: Evading WAF

• Handcrafted grammar with valid suffixes of SQL statements.
  - SELECT * from table WHERE id=S
  - Simulates an SQL Injection attack.

• Test GOFA algorithm against live installations of ModSecurity and PHPIDS.
  - Both systems include *non regular* anomaly detection components.
GOFA: Evading WAF

Evasions found for both web application firewalls.

 ✓ **Authentication Bypass:** 1 or isAdmin like 1

 ✓ **Data Retrieval:** 1 right join users on author.id = users.id

Evasion attacks acknowledged by ModSecurity team.
Conclusions

• SFAs provide an efficient way to infer regular expressions.

• SFA learning can provide insights for *non regular systems*.

• Similar techniques derived for sanitizers, more in the paper!

• Large space for improvements over presented learning algorithm.
  - Smarter guard generation algorithms.

• We envision **assisted** Black-Box testing of sanitizers and filters.
  - Auditor will correct inaccuracies of models.
  - Derive concrete attacks from abstract language constructs.
Back In Black:
Towards Formal, Black Box Analysis Of Sanitizers and Filters

George Argyros*, Ioannis Stais**, Angelos Keromytis* and Aggelos Kiayias***