Operating System Discovery Using Answer Set Programming

François Gagnon
PhD Student
Carleton University
fgagnon@scé.carleton.ca
www.sce.carleton.ca/~fgagnon

Current OSD Approaches

Passive:
• Lack knowledge representation:
  • One guess per packet
  • No memory of previous guesses
  • Limited to the information they receive
  • Limited accuracy

Active:
• Lack knowledge representation:
  • Redo the work for each query
  • No memory of previous test results
  • Lack planning ability
  • Always run all tests
  • Very noisy
  • Don’t use the information freely available

What is Operating System Discovery?
• Remotely identifying which operating systems are running on distant computers

Operating System Discovery Using Answer Set Programming

What is Operating System Discovery?
• Remotely identifying which operating systems are running on distant computers

Answer Set Programming

• Extended disjunctive logic programs with answer set semantics

  \[ L_1 \lor \ldots \lor L_k \lor \neg L_{k+1} \lor \ldots \lor \neg L_m, \neg L_{m+1} \lor \ldots \lor \neg L_n \]

  • Where each \( L_i \) is a literal (\( A \)) or its strong negation (\( \neg A \))
  • \( \neg \) denotes weak negation

• A set of ground literals \( S \) is an answer set of program \( \Pi \) if:
  • the literals of \( S \) are those made true by \( \Pi \)
  • the literals of \( S \) are sufficient to respect the rules of \( \Pi \)
  • no proper subset of \( S \) is also an answer set

• A program may have multiple answer sets
• The language is fully declarative (can be generated automatically)

Benchmarks (POSD)

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>6671 attack traces containing up to 63000 packets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>85 different OS</td>
</tr>
</tbody>
</table>
| Time     | avgTime: 300 ms
          | minTime: 100 ms
          | maxTime: 10623 ms

Passive Module (POSD)

• Represented as an explanatory diagnosis problem \( \text{hyp}(\text{th}, \text{obs}) \)

  • \( \text{hyp} \): the set of currently possible explanations (OS)
  • \( \text{th} \): a set of rules describing the behavior of each OS
  • \( \text{obs} \): the packets seen so far

• Example of a behavior rule:

  \[ \text{os}(\text{win2K}) \lor \text{os}(\text{winXP}) \lor \text{tcp}(\text{IP}S,\text{IP}D,\text{PS},\text{PD},\text{yes},\text{syn},128) \]

• Each answer set provides a possible explanation (OS)
• We prioritize smaller answers since they are more general
• If no answer set contains only 1 OS, it could mean:
  • The target changed its OS
  • The target is actually multiple computers behind a NAT
  • The observations non-monotonically confirm some hypotheses
• The logic program is generated automatically

Active Module (Future Work)

• Given a set of hypotheses \( H \), generated by the passive module, we can ask several questions:
  • Is \( O \) the actual operating system?
    • \text{yes} if \( H = \{O\} \)
    • \text{no} if \( O \not\in H \)
    • \text{unknown otherwise}
  • Does the actual operating system belong to \( \theta \)?
    • \text{yes} if \( \theta \supseteq H \)
    • \text{no} if \( \theta \cap H = \emptyset \)
    • \text{unknown otherwise}
  • What is the actual operating system?
    • \text{if} \( H \) is a singleton \( \{H\} \)
    • \text{unknown otherwise}
  • Generate a (conditional) plan to gather the missing observations