Machine Learning for Implanted Malicious Code Detection

Yating Hsu (OSU)

David Lee (HP Labs, LOA from OSU)

FIST 2011

Outline

- Problem
- Structured Analysis and Machine Learning
- Incomplete Specified Machine
- Experiments

Motivation

- UnrealIRCd 3.2.8.1
  - IRC server
  - Replaced with a version with backdoor on minors sites
  - Allow remote execution of arbitrary system commands
- Vsftpd 2.3.4
  - FTP server
  - Replaced with a version with backdoor on master site
  - Launch a TCP callback shell when login with “:)”

Problem Statement

- Given a black box protocol implementation, we want to determine
  1. If there is any malicious behavior hidden in the black box
  2. The function of the hidden code if any

- Proposed approach: synthesize a behavior model of the protocol implementation

Proposed Method

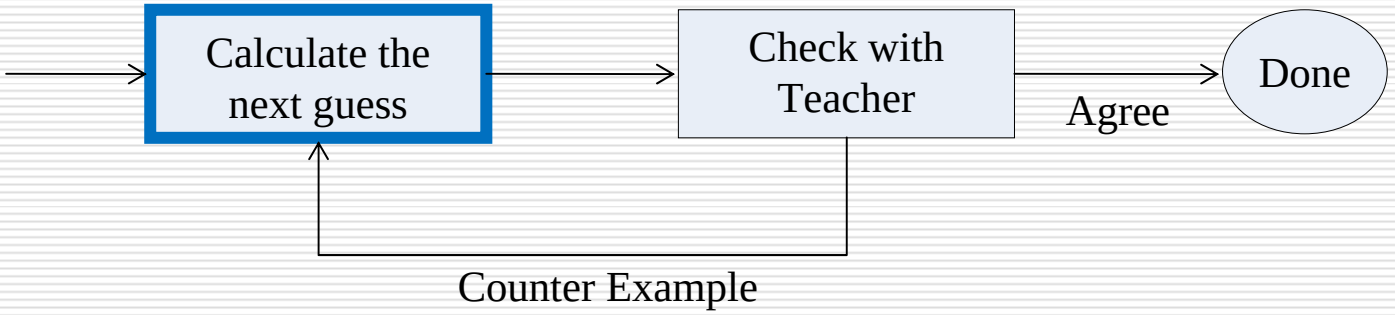
- Given a protocol implementation possibly implanted with malicious code
- Obtain the input/output message format through reverse engineering tools
  - Polyglot, AutoFormat, REWARDS, Tupni, Dispatcher
- Synthesize a behavior model to represent the protocol implementation
  - Generalize *L** algorithm
- Examining the model to understand hidden behavior

Finite State Machine Model

- Finite State Machine (FSM)
  - $M = (I, O, Q, \delta, \lambda)$
    - I=input symbols, O=output symbols, Q=states, δ=transition function, λ=output function
  - Fully specified FSM
    - At each state upon any input, there is a specified next state and a specified output
    - Otherwise, an incompletely specified machine
- FSM Machine Learning
  - Synthesize an FSM to represent the behavior of the given black box

Machine Learning: *L** Algorithm

- Maintain a guess (Observation Table) of the black box
- Assume the existence of a *Teacher*
  - Teacher can answer two type of queries
    - Output query
      - Given an input string, ask “what’s the output”
    - Equivalence query
      - Ask “is the current guess the same as the black box”
      - Teacher give counter example if not
- Learning procedure

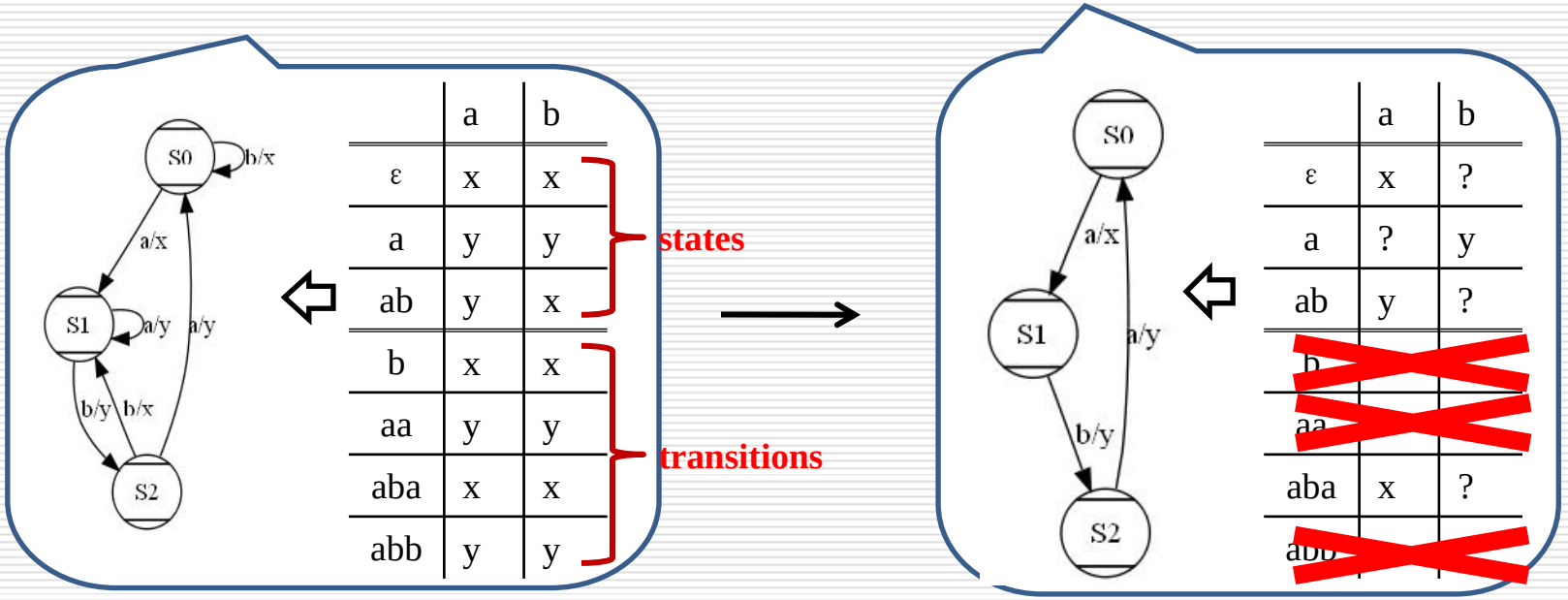


*L** Algorithm

- Limitation of *L**
  - Learn fully specified machine
    - Not all inputs are expected at each state
    - Malicious code are based on inputs, which are not specified in system normal operations
  - Start from empty Observation Table
    - We may have information of the normal behavior

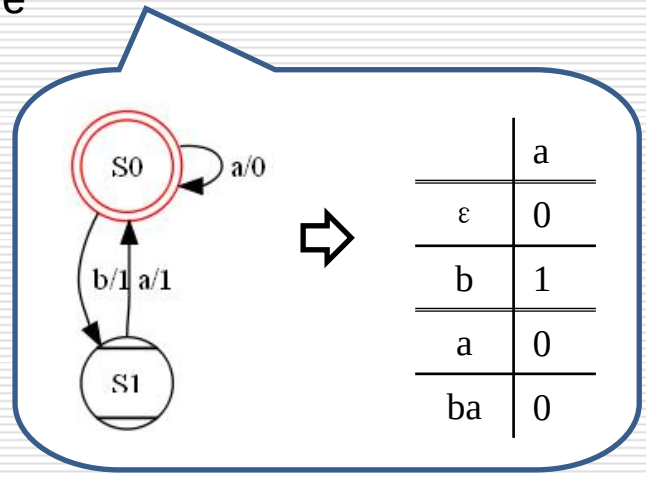
Generalized L^*: L_{isfsm}* (1)

- Supervised FSM Learning
- Generalize L^* to learn incomplete specified FSM
  - Not all inputs are expected at each state
  - Modify the definition of consistency and closeness
  - Maintain smaller observation table



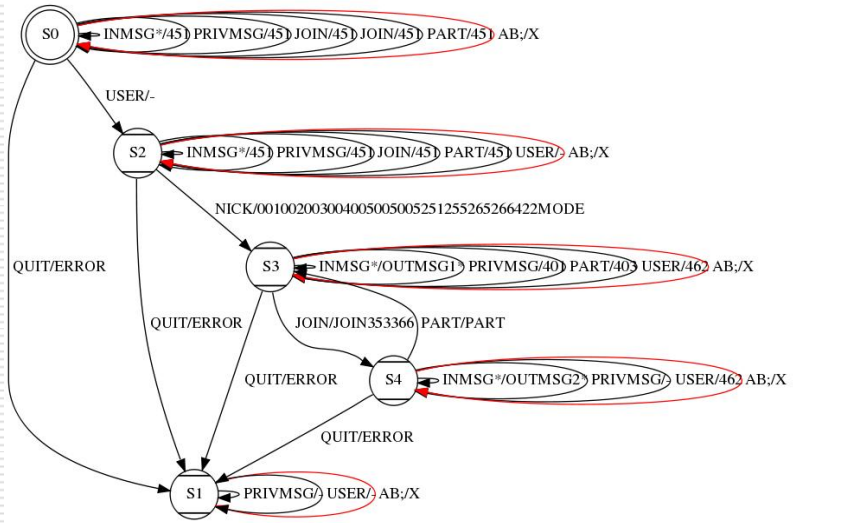
Generalized L^*: L_{isfsm}^* (2)

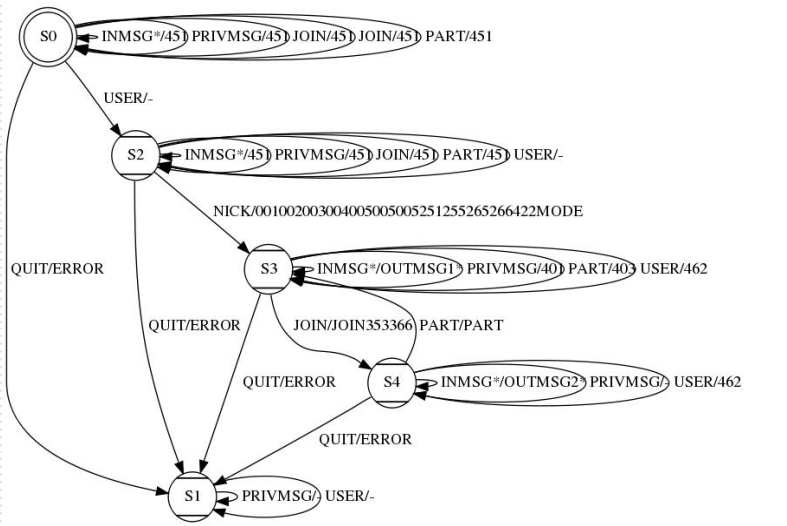
- Modify learning procedure to start with an FSM
  - Implanted malicious code detection
    - The model of the normal behavior M_0 may be available
      - Specification, development manual …
      - May be identical to implementation
  - Translate M_0 to an Observation Table OT_0
    - Rows: shortest input sequence from initial state to each state
    - Columns: separating sequence
    - Outputs fill the table
  - Speedup learning process



Experiment

- UnrealIRC 3.2.8.1
  - IRC server with backdoor
  - Input alphabet
    - Normal IRC messages
    - “AB;” followed by any system command





Experiments

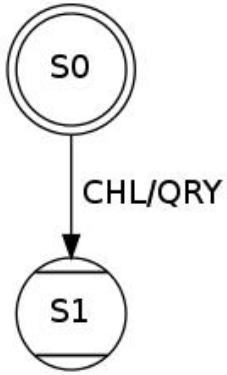
- UnrealIRC 3.2.8.1
  - Observation table: save 23.5% of space
  - Start from empty observation table: 1213 queries
    Start from clean IRC black box: 409 queries

Experiment

- MSN client with message flooder
  - Input alphabet
    - Only normal MSN messages, malicious function does not introduce extra I/O
  - MSN client

w/ message flooder

w/o message flooder



Conclusion

- Generalize the classic machine learning algorithm
- Automatically discover hidden behavior inside protocol implementations with machine learning
- Self-taught learning