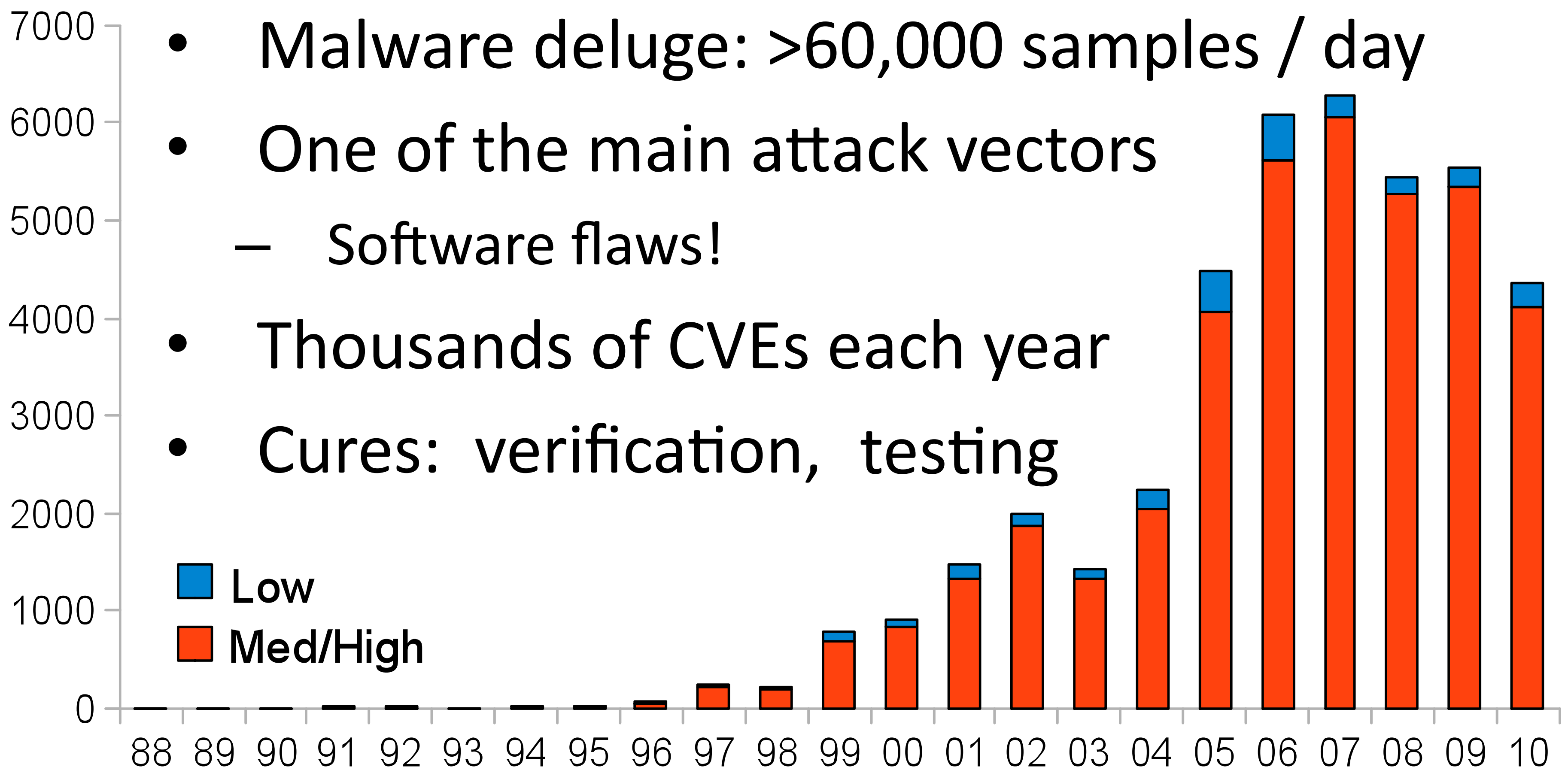
MACE:

Model-inference-Assisted Concolic Exploration

Domagoj Babic
http://www.domagoj.info/
joint work with Chia Yuan Cho, Pongsin Poosankam,
Kevin Zhijie Chen, Edward XueJun Wu, Dawn Song
UC Berkeley

Software Security



- Malware deluge: >60,000 samples / day
- One of the main attack vectors
  - Software flaws!
- Thousands of CVEs each year
- Cures: verification, testing

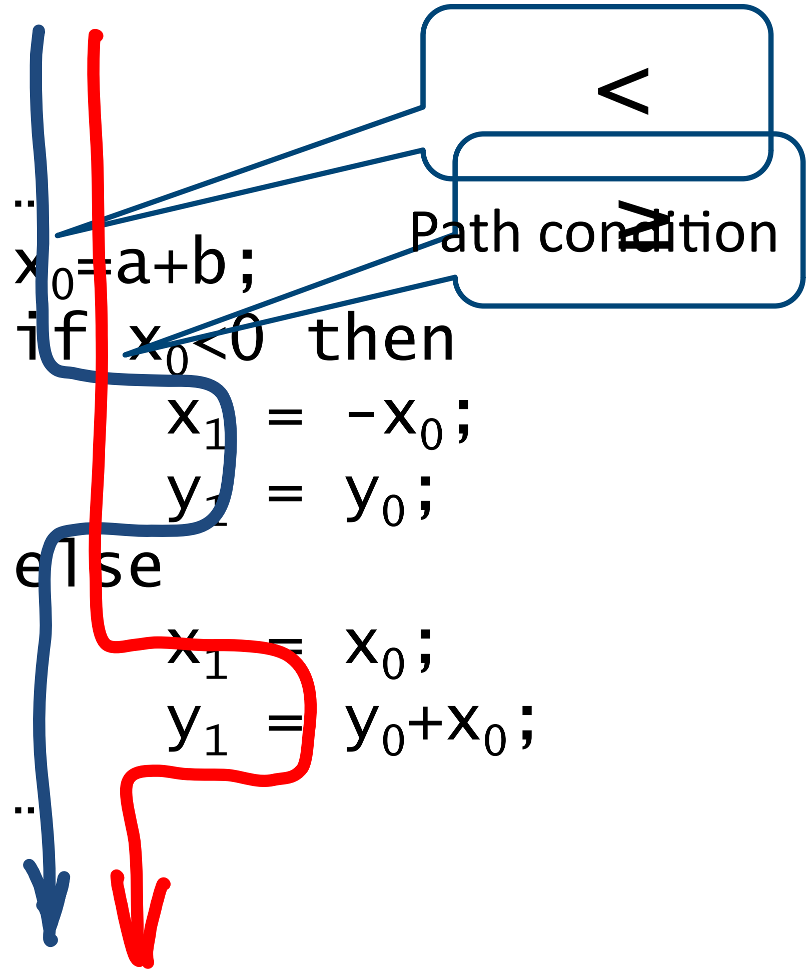
Common Vulnerability Disclosures (CVEs) per year

Outline

- Dynamic Symbolic Execution (a.k.a. DART, concolic execution)
  - High-level intro
  - Aspects that could be improved
- Model-inference-Assisted Concolic Exploration
  - How it works
  - How it improves over dynamic symbolic execution
- Experimental results

Dynamic Symbolic Execution

- Independently invented by several groups in 2004/2005
- Main components:
  - Concrete execution
  - Symbolic execution
  - Solver (decision procedure)
- Very effective in practice



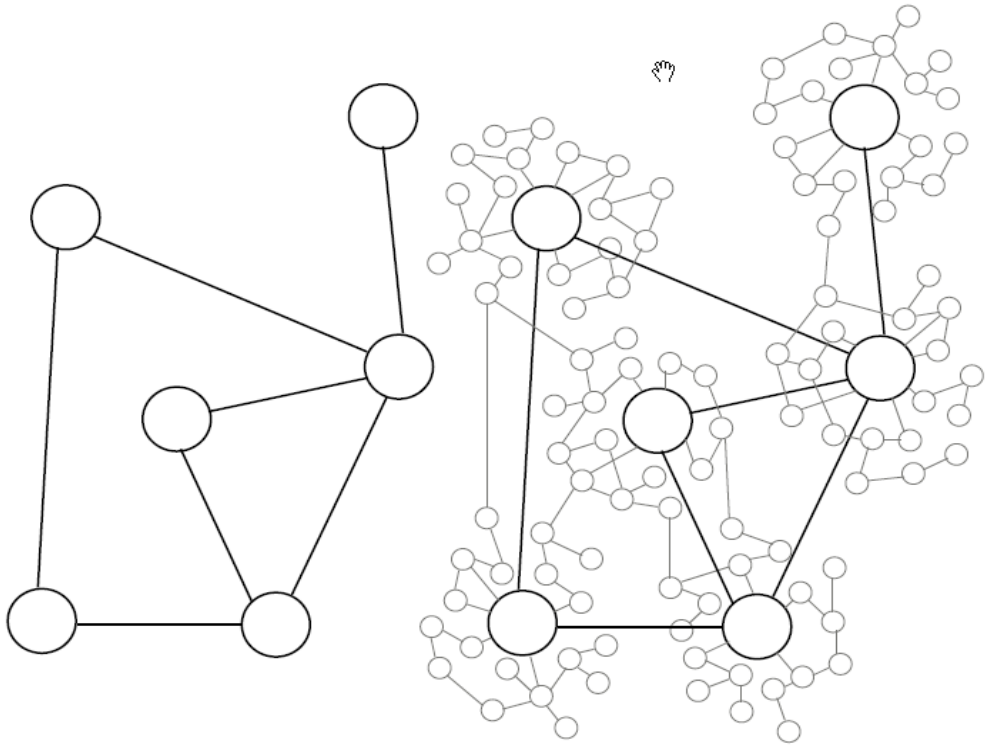
```
..
x0=a+b;
if x0<0 then
    x1 = -x0;
    y1 = y0;
else
    x1 = x0;
    y1 = y0+x0;
..
```

Learning

- Dynamic symbolic execution
  - Repeats iterations (concrete + symbolic) until terminated
  - Knowledge gained from iterations discarded
- Research questions:
  - What can be learned from iterations?
  - How can one represent the gained knowledge?
  - How could that knowledge prune the search space?

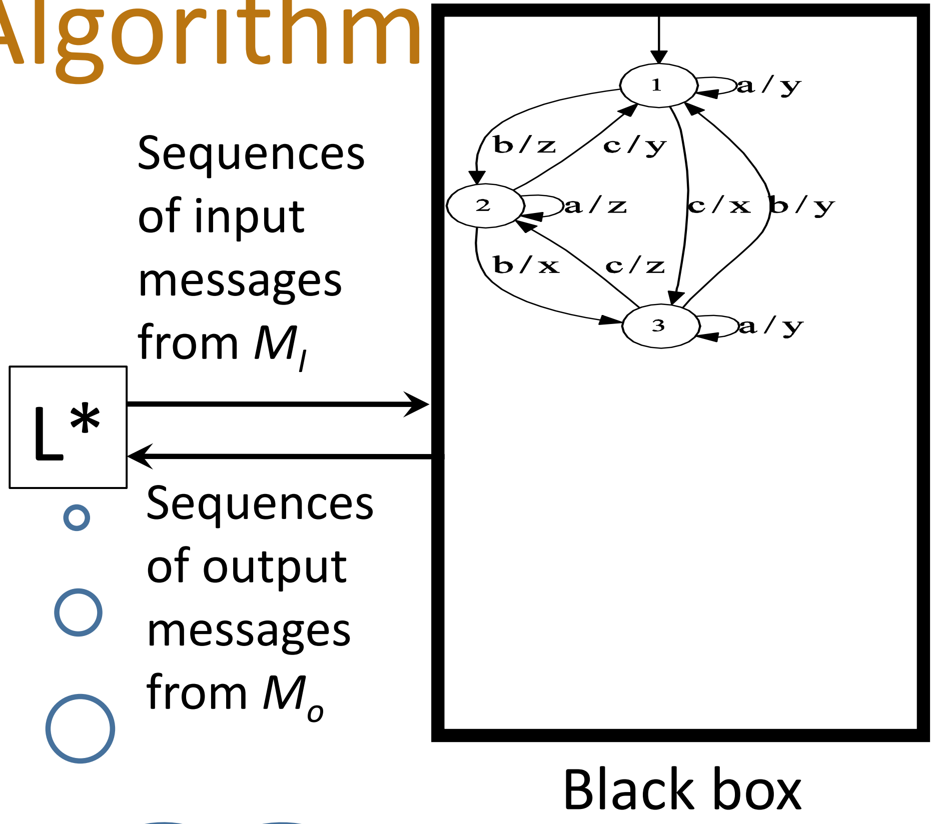
MACE – The Main Ideas

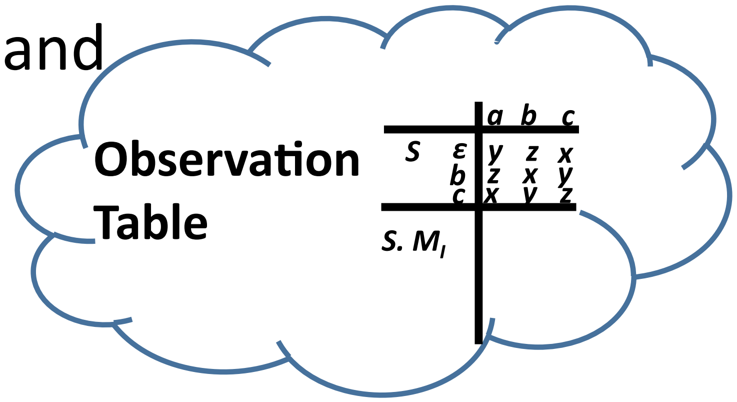
- Learning + dynamic symbolic execution
- Learns a state-machine abstracting the program
  - Guides further search
    - Initialize the program to certain state
    - Explore the neighborhood
  - Specifies sequences of inputs required to get to a certain state



The L* Algorithm

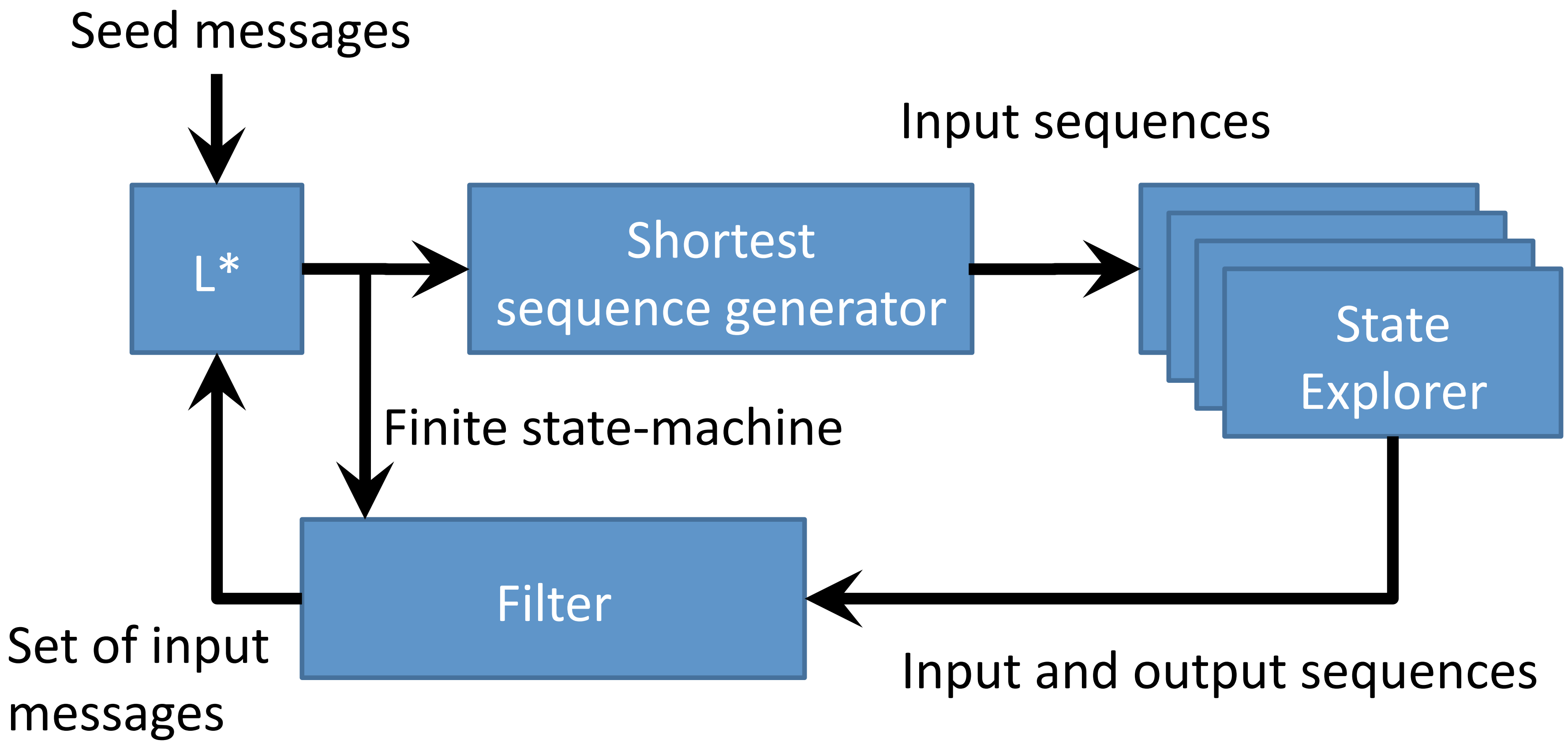
- MACE uses an improved L* [CCS'2010]
- Polynomial in the number of states and size of the input message set M_I
- Constructs an *observation table*
- Reads off states and transitions from the table


Sequences of input messages from M_I
L*
Sequences of output messages from M_o
1
2
3
a/y
b/z
c/y
a/z
c/x
b/y
b/x
c/z
a/y
Black box


Observation Table

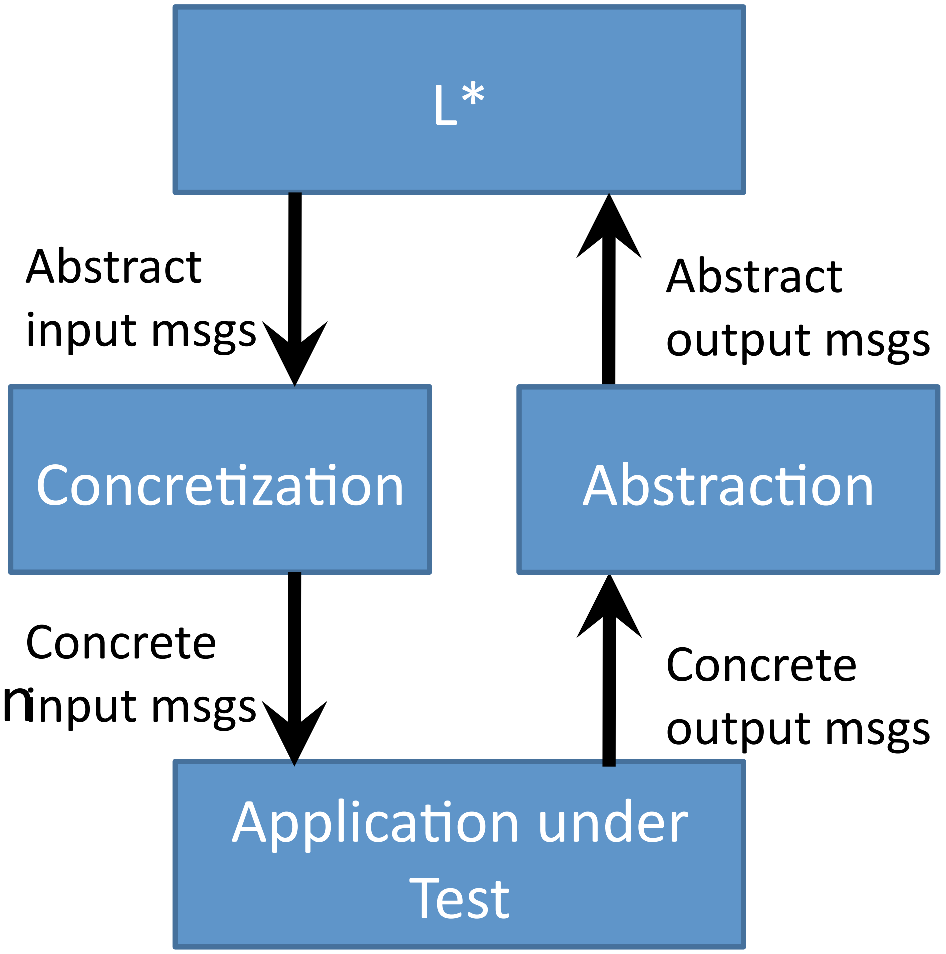
		a	b	c
S	ε	y	z	x
	b	z	x	y
	c	x	y	z
$S.M_I$				

The MACE Approach



Key Difficulty: Abstraction of Messages

- Inferring the state-machine over all messages
  - Computationally infeasible
  - Useless for guidance
- L* operates over an abstract set of messages
- In prior work [CCS'10] – manually written abstractions
- MACE: automatic abstraction of input messages



Filtering Function

- The main idea: keep only the messages that refine the state-machine
- Exact check too expensive, use an approximation
  - If the current state-machine can produce the given output sequence, no refinement
  - Otherwise, add all the input messages from the corresponding input sequence

$$A \times M \quad \times M \quad \rightarrow$$

$$= \left\{ \quad \exists \in \quad \lambda \quad = \right.$$

Implementation

- Dynamic symbolic execution engine
  - BitBlaze infrastructure
- L*
  - Our implementation with improvements from the CCS'2010 botnet analysis paper
- Scripts
  - For gluing the components together

Applications of MACE

- Guiding dynamic symbolic execution
  - Different abstractions suitable for different types of applications
  - E.g., inference of context-free grammars for automated testing of applications with parsers
- Protocol reverse engineering
  - Comparative analysis (e.g., for extracting signatures)
  - Protocol state-machine model checking

Experimental Setup

- DETER Security testbed
  (3GHz Intel Xeon processors)
- State-space exploration done in parallel
  - One job per state in the inferred state-machine
  - 2.5 hr timeout per state
  - Each newly discovered state explored only once
- For coverage measurement experiments
  - Baseline got extra time, compensates for the time spent in learning

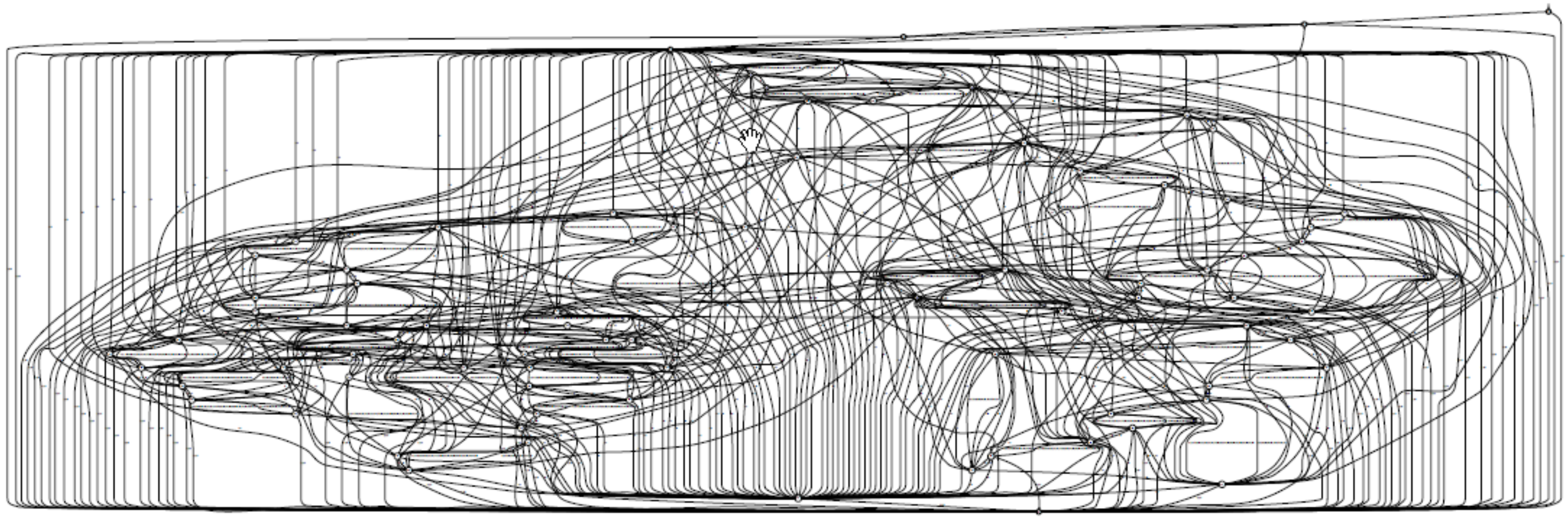
Benchmarks

- Inference done on
  - Remote Frame Buffer (RFB) protocol: Vino 2.26.1
  - Server Message Block (SMB) protocol: Samba 3.3.4
- State-space exploration also done on
  - RealVNC
  - Win XP SMB
- Seed message set
  - Vino: 45 sec session of a remote desktop session
  - Samba: used `gentest` suite

Results: Iterations and Runtime

Program	Iteration	States	Input alphabet size	Output alphabet size	Learning time (min)
Vino	1	7	8	7	142
	2	7	12	8	8
Samba	1	40	40	14	2028
	2	84	54	24	1840
	3	84	55	25	307

Results: Inferred Protocol Models



Inferred 84-state SMB protocol implementation abstraction

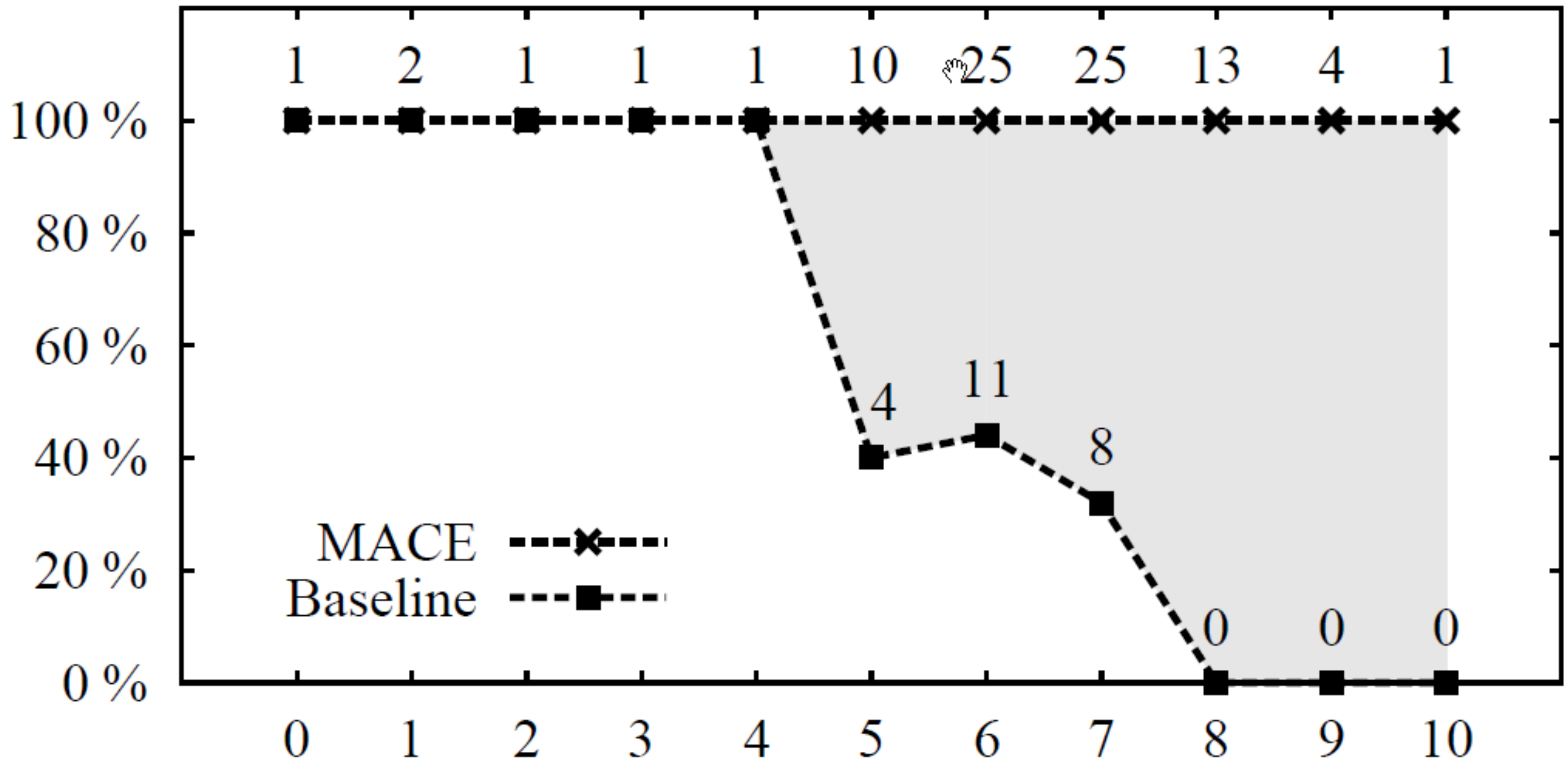
Results: Discovered Vulnerabilities

Program	Vulnerability	New	MACE (hrs)	Baseline (hrs)
Vino	CVE-2011-0906	✓	1	N/A
	CVE-2011-0905	✓	4	>105
	CVE-2011-0904	✓	15	>105
Samba	CVE-2010-2063		12	602
	CVE-2010-1642		14	>1260
	Fixed without CVE		124	>1260
RealVNC	CVE-2011-0907	✓	2	>105
Win XP SMB	None		>210	>1260

Results: Coverage Improvement

Program	Instruction Coverage Baseline	Instruction Coverage MACE	Coverage Improvement (%)
Vino	129762	138232	6.53
Samba	66693	105946	58.86
RealVNC	39300	47557	21.01
Win XP	90431	112820	24.76

Results: Exploration Depth (SMB)



Why MACE Works so Well?

- Uses a relatively cheap technique (L*) to infer an abstraction of the search space and reduce the search space
- The abstraction is used to guide the search
  - Especially useful for constructing sequences of messages to get to certain state
- More control over the search
  - E.g., decreases the probability of getting stuck in loops

Summary

- Model-inference-Assisted Concolic Execution
  - How it works
  - How it improves dynamic symbolic execution
- Experimental results
  - 7X more vulnerabilities found
  - Up to 58% better coverage
  - Deeper states explored