Evolutionary Computation for Improving Malware Analysis

Kevin Leach[1], Ryan Dougherty[2], Chad Spensky[3], Stephanie Forrest[2], Westley Weimer[1]

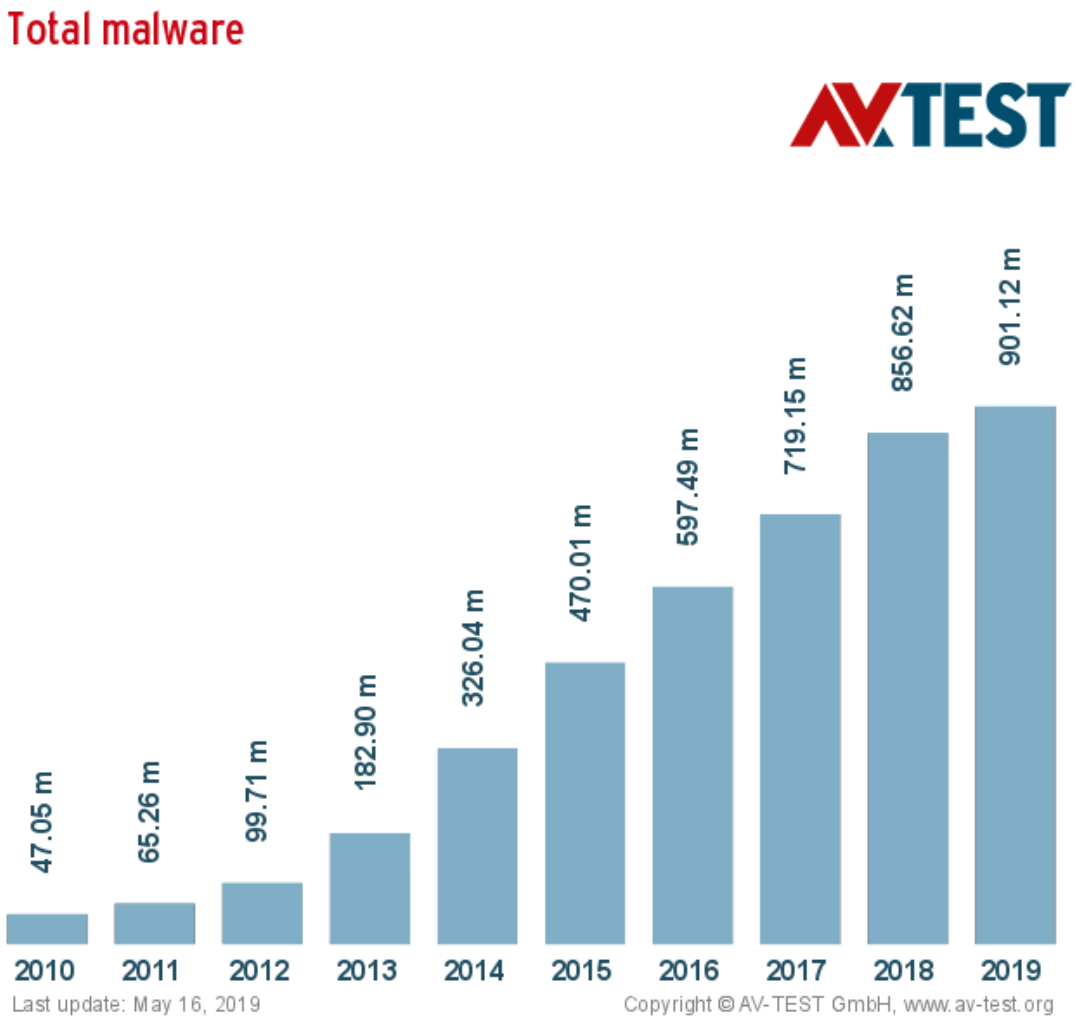
[1]University of Michigan
[2]Arizona State University
[3]University of California, Santa Babara

June 1, 2019

Setting the Stage

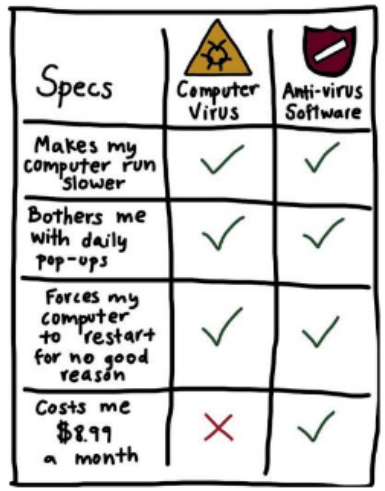
- Can we improve the *efficiency* of automated malware analysis using evolutionary computation?

Introduction



Malware Analysis

- Analysts want to quickly identify malware behavior
  - What damage does it do?
  - How does it infect a system?
  - How do we defend against it?

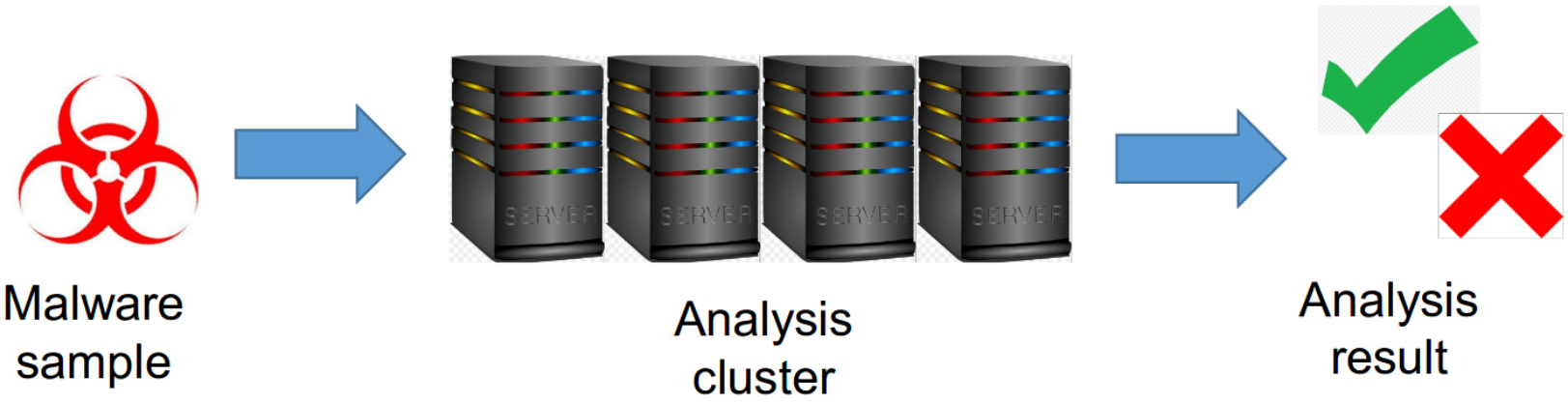


Stealthy Malware

- Growing volume of *stealthy* malware
- Malware sample maintains secrecy by using *artifacts* to detect analysis environments
  - *Timing artifacts* — overhead introduced by analysis
    - Single-stepping instructions with debugger is slow
    - Imperfect VM environment does not match native speed
  - *Functional artifacts* — features introduced by analysis
    - `isDebuggerPresent()` — legitimate feature abused by adversaries
    - Incomplete emulation of some instructions by VM
    - Device names (hard drive named “VMWare disk”)
- **Automated analysis is difficult**

Automated Malware Analysis

- Cluster of servers analyzes malware
  - Analysis success depends on cluster's environment (e.g., OS, virtualization)



Transparency

- We want to understand stealthy samples
- We can *mitigate* artifacts
  - Hook/intercept API calls
    (e.g., `isDebuggerPresent()`)
  - Spoof timing
    (e.g., virtualize result of `rdtsc` instruction)
  - Use alternate virtualization
    (e.g., a sample that detects VMWare may not detect VirtualBox)

Cost of Transparency

- Mitigation **costs** resources
  - Development effort
    (e.g., modifying virtualization)
  - Execution time
    (e.g., due to runtime overhead)
- Mitigation **covers** some subset of malware
  - Artifact category
    (e.g., hooking disk-related APIs covers malware that checks the disk)

Key Idea: Transparency/Cost

- We can **control** which artifacts are exposed
  - i.e., control costs and coverage
- Use a vector of yes/no answers

VMWare?	large disk?	Spoof timers?	cost
0	1	0	x
1	0	1	y

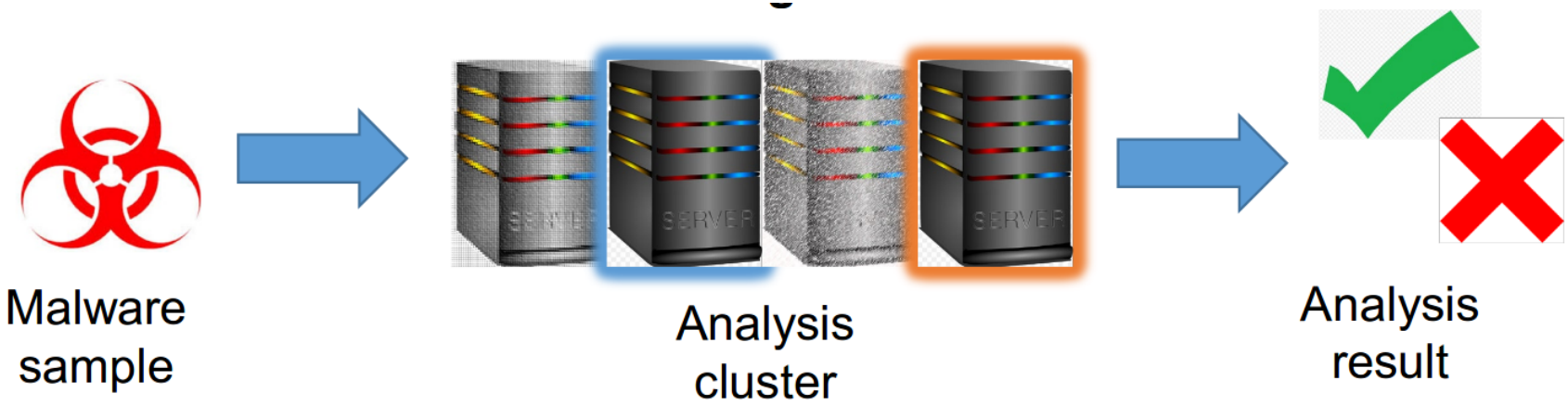
Key Idea: Transparency/Cost

- Derive a **cost model** empirically
  - What are the values of x and y?

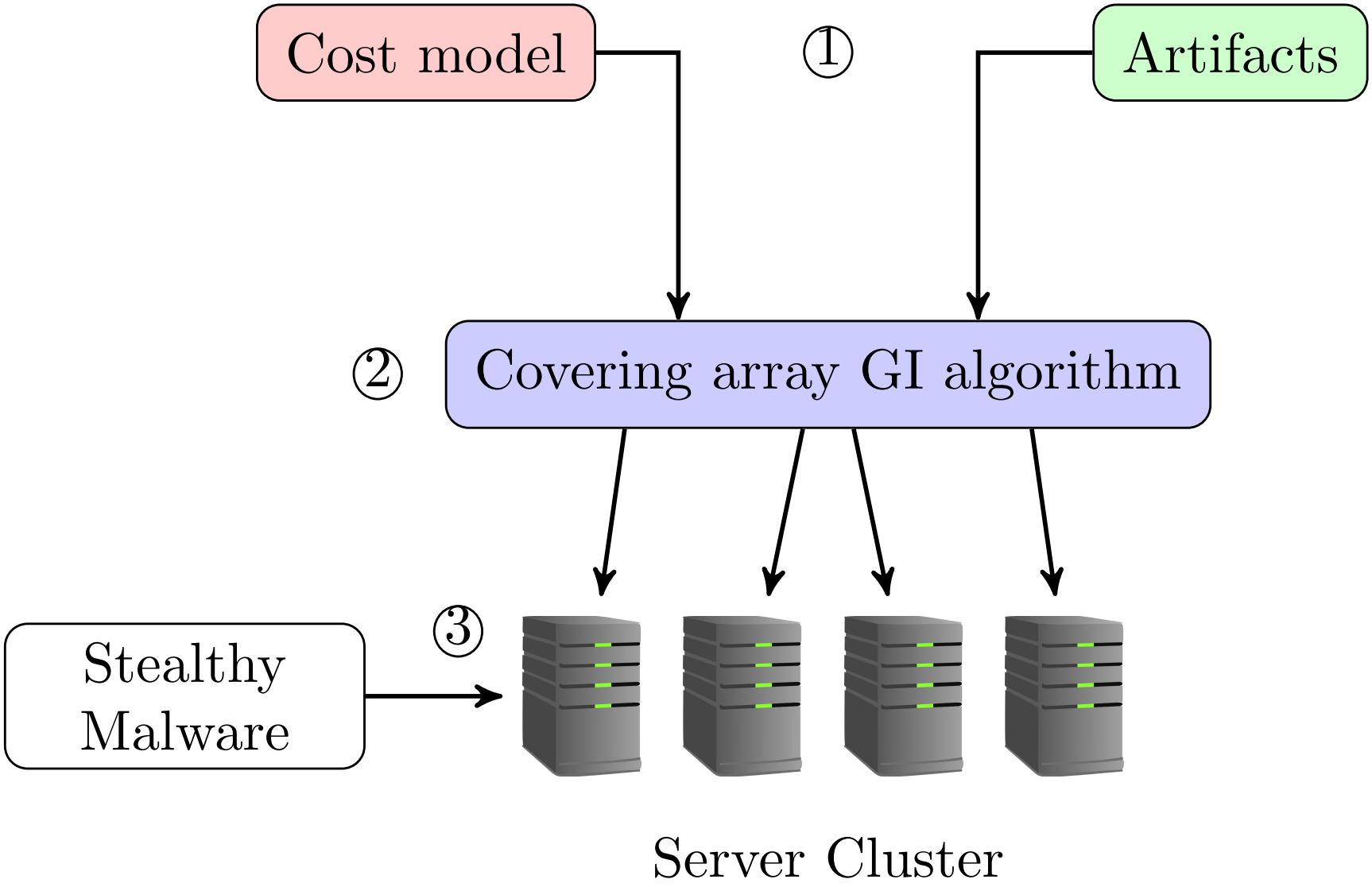
VMWare?	large disk?	Spoof timers?	cost
0	1	0	x
1	0	1	y

Automated Malware Analysis

- Can we analyze more stealthy malware if we offer different analysis environments?



Proposed Architecture



Proposed Approach

Given: Cost model, number of servers
Find: settings that minimize cost, maximize coverage

Config	Server	A0	A1	A2	cost
1	1	0	1	0	3
	2	1	0	1	2
2	1	0	1	1	4
	2	1	1	0	3
3	1	0	0	0	1
	2	1	1	1	7

Proposed Approach

Given: Cost model, number of servers
Find: settings that minimize cost, maximize coverage

Config	Server	A0	A1	A2	cost
1	1	**0**	**1**	**0**	3
	2	**1**	**0**	**1**	2
2	1	0	1	1	4
	2	1	1	0	3
3	1	**0**	**0**	**0**	1
	2	**1**	**1**	**1**	7

Proposed Approach

Given: Cost model, number of servers
Find: settings that minimize cost, maximize coverage

Config	Server	A0	A1	A2	cost
1	1	0	1	0	3
	2	1	0	1	2
2	1	**0**	**1**	**1**	4
	2	**1**	**1**	**0**	3
3	1	0	0	0	1
	2	1	1	1	7

Initial Results

- **Challenge** Ground-truth data is hard to come by
  - Manually reverse engineered 20 samples
- **Baseline** analyze each with high-transparency, high-cost analysis (e.g., all 1's)
  - roughly 360x overhead
- **Cost Function** We estimated a cost function based manual review

Initial Results

Baseline	Us / 2 servers	Us / 4 servers	Us / 8 servers
360x	16.14 – 64.51	12.06 – 49.39	1.05 – 30.92

- Promising improvements to **throughput** (potentially, by 1–2 orders of magnitude)
- **Future work**
  - Ongoing analysis involving 20k+ stealthy samples
  - Need to empirically derive cost function (rather than manual assessment)

Conclusion

- We can **control** which **artifacts** are exposed to **stealthy malware samples**
- We can **evolve** a set of analysis server **configurations** to maximize **coverage** while minimizing **cost**