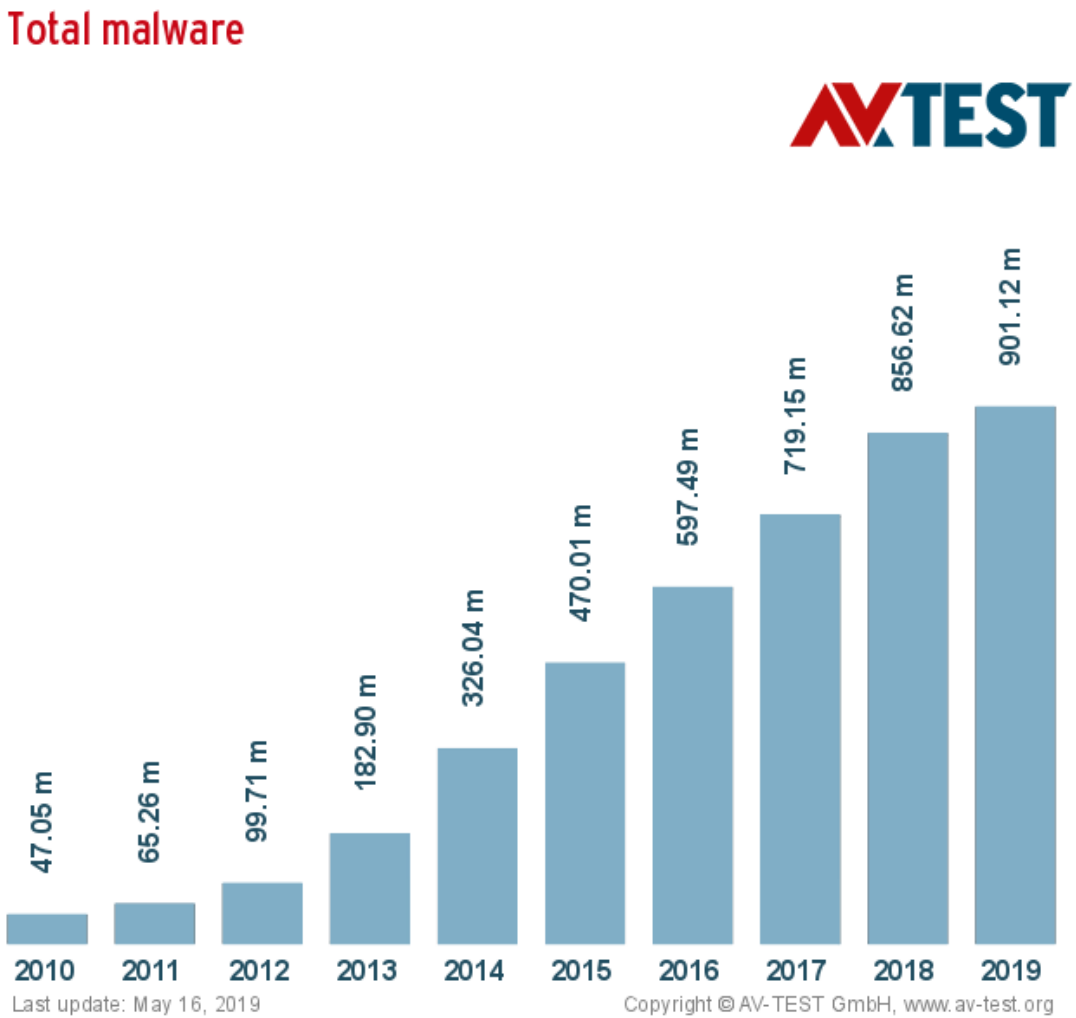
Evolutionary Computation for Improving Malware Analysis

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

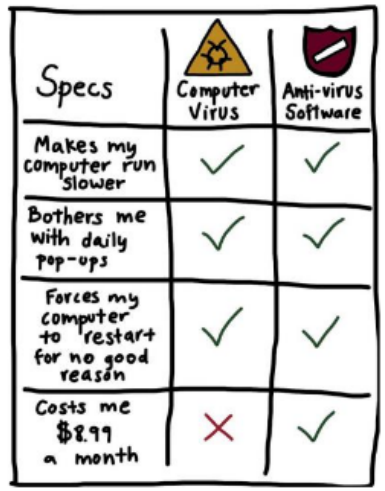
May 23, 2019

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”)
- **Too much effort to analyze**

Transparency

- We want to understand stealthy samples
  - We want a *transparent* analysis
- We can *mitigate* artifacts
  - Hook 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 takes **resources**
  - Development effort
    (e.g., modifying virtualization)
  - Execution time
    (e.g., due to runtime overhead)

- Mitigation **covers** some subset of malware
  - Artifact category
    (i.e,. hooking disk-related APIs covers malware that checks the disk)