AV-TEST
Tests of Anti-Virus-Software independent • qualified • fast

Testing of “Dynamic Detection”

Maik Morgenstern, Andreas Marx
AV-Test GmbH, Magdeburg, Germany

http://www.av-test.org

Presented at the AVAR 2007 Conference in Seoul, South Korea
http://www.aavar.org/avar2007/index.html

Table of Contents

- Before: Static detection as part of the “traditional” way of AV testing
- Now: Dynamic detection and how to test it
  - Dynamic detection
  - Ideal test setup
  - Basic things to consider
  - Problems and solutions
  - Concluded test setup
  - Some real testing experiences

History (I)

- Static detection
  - Signatures and heuristics used for detection
  - Check files one by one
  - Malware is detected before/without execution
- Traditional AV testing
  - Test for detection of samples
  - Samples sorted into different sets, depending on the type of the malware
  - Zoo and WildList malware
    - Usually old and “outdated” samples are tested
  - On-demand an on-access tests

History (II)

- Already many extensions to types of testing
  - Retrospective testing
  - Outbreak response times
  - Malware disinfection testing (viruses)
  - System cleaning and malware removal testing (e.g. worms, bots, backdoors)
- But none address dynamic detection, which is becoming an important part of security software
  - Our AV lab is receiving between 2,000 and 2,500 new unique malware samples per hour!

Dynamic Detection (I)

- Security application:
  - Reviews the software's behavior
  - Decides whether it is malware or not, depending on how it arrives (introduction vector) and what it does
  - Doesn't just review an isolated component: instead, a whole system of components and behavior is assessed

Dynamic Detection (II)

- Security software:
  - Blocks detected malicious actions on detection (or asks whether to block or not)
  - Kills the process that performs the malicious actions and handles the executable on disk (or asks whether to do so or not)
  - Reverses changes made by the malicious program (or asks …), Reversal includes removal of additional components that have been created

Dynamic Detection (III)

- Testing dynamic detection:
  - Simulation tools or real malware
  - Set of malware samples has to be determined
  - Have to review what malware does as well as what the security application does
  - Establish an isolated environment that provides the functionality needed for the malware to “work” without risk to external environment
  - Measure success in terms of detection and blocking

AV-TEST
Tests of Anti-Virus-Software independent • qualified • fast

“Ideal” Test Setup (I)

1. Use real (not brand new, not outdated) hardware. (Virtual machines may be used to compare results to results on real machines.)
2. Base system runs recent OS version (latest SP?)
3. Patch level should reflect the desired scenario (testing for vulnerability to exploits)
4. Products will (subject to test spec) be tested with the default settings
5. Use malware samples not detected by signatures (if only testing proactive detection)

“Ideal” Test Setup (II)

6. Should use high volume of malware samples and cover many different malware types
    a. Real malware as currently found in the wild will be used (no artificial setups: real e-mails, malicious websites and so on are used as sources)
    b. If (a) not possible use a perfectly simulated internet with recent samples (introduced and executed as they would be in real life)
7. Introduce the malware sample via the desired and appropriate introduction vector (if the infection vector - e.g. e-mail, web pages, download - is known)
8. Record the impact of the security software and compare the result to the actions of the malware on the clean base system
9. Assess success in detection, reporting and blocking

Measurement of Success (I)

State of malware	Detection	Comments for testing
Arrival on the system	Identify exploits, malicious packets, malicious web source etc.	Easy to test: either malware arrives on the system or it doesn't
Before execution	Signature-based (or other static) detection	Easy to test: either malware is detected and the execution is blocked or it isn't
After execution a) Before doing damage b) After doing damage	Behavior-based (dynamic) detection	Harder to test, since malware is executed and might perform actions on the system that have to be reviewed one by one

Measurement of Success (II)

- Pre-Requisite Information:
  1. The clean state of the operating system has to be known.
  2. The changes of the malware to the system and other actions have to be known (when there is no security software installed)
     a. The modifications and behavior should also be reviewed after a reboot (since some malware either doesn't survive reboots or exhibits different behavior when it executes after a reboot)
  3. The state of the operating system after the malware has been executed in the presence of the tested security software

Measurement of Success (III)

- Detection
  - Any messages (informational vs. query/interactive) displayed by the program?
- Blocking and reversing malicious actions
  - Are malicious actions blocked automatically or is the user asked to decide? (Are both options available?)
  - Which malicious actions have been blocked or reversed? (All? Is system returned to pre-infection state?)
  - Which components of the identified malicious program have been removed?
  - How are the different actions and components to be rated regarding risk?

False Positives and Noise (I)

- There are several approaches to involving the user in the blocking process:
  - Software may ask to block or reverse actions completely because of suspicious behavior
  - Interaction/confirmation requested only for certain actions, because they look suspicious
  - There are no questions at all and the security software silently does the job or only notifies the user, without asking for confirmation
- What's the best approach?

False Positives and Noise (II)

- Messages are important: the user wants to know what's going on and what actions he has to react to
- However too many messages, questions and requests for confirmation can be unpleasant and have a negative effect
- The user might get used to the messages and simply "learns" to click them away without reading them anymore
- The number and type of messages during the tests should be determined → Noise Tests

False Positives and Noise (III)

- There is also the problem of false positives (FPs)
- Clean software or the actions of clean software are blocked or reported as malicious
- When this happens, there should at least be the option to explicitly allow the blocked software or actions

False Positives and Noise (IV)

- False positive test setup
  - Mainly, use multiple versions of widespread standard applications like Microsoft Office, OpenOffice, Mozilla Suite, Adobe Reader, messenger software, legit toolbars, media players; perform a Windows Update
  - To cover as many bases as possible, specialized and rarely used applications can be included as well
  - Go through the whole installation process as well as (built-in) updating mechanism, verify functionality of the FP suite
  - Note messages displayed by the security software, possible choices to block/allow and the impact of these choices on the functionality of the tested application

Performance Testing

- Security software can have quite a significant impact on the system performance, especially where dynamic detection is in use, because of the live monitoring of applications
- Benchmark tools as WinStone or Sysmark might be used (Be careful, sometimes these tools behave unpredictably; and don't forget to disable the included security software tests, if present.)
- Or perform tests manually: measure boot and shutdown time, launch time of multiple applications
- First measure timing without any security software installed as a baseline/reference, then perform the same tests with security software installed, and repeat it 3+ times
- The smaller the difference, the lower the impact

Obstacles to the “Ideal” Testing Strategy

- As the term “ideal” implies, this testing strategy can’t be applied in reality without making some changes
- There are many variables and options, all of which might influence malware behavior
- Review these variables and obstacles and suggest solutions and workarounds
- Base approaches and options will be worked out

Hardware and Operating System (OS)

- Base (default) approach:
  - Real hardware
  - Required operating system with latest service pack but without hotfixes (at the moment Windows XP SP2 or Windows Vista SP0)
  - US-English as language
  - Default settings of the operating system, except to disable OS-included security solutions as Microsoft Windows Defender on Vista
- Options:
  - Virtual machines as alternative or comparison (misdetection should be validated under real hardware)
  - Other operating systems, depending on the customers needs, with the pre-agreed service packs and hotfixes
  - Language of choice
  - Special settings of the operating system as desired, i.e. non-default

Configuration of the Security Software

- Base approach:
  - Test the security products with default settings
  - Update products to their latest version
  - Turn off signature-based detection or freeze signatures for a certain amount of time until choosing samples and starting the test
  - Make sure products can’t update themselves during the test
- Options:
  - Change settings to other states (e.g. best possible settings)
  - Use products off the shelf without any updates
  - Instead of freezing signatures, roll them back manually
  - Use different freeze points

Implementation of Test (I)

- Base approach:
  - Choose samples, consider constraints regarding signature detection
  - The number of samples should be statistically significant and a wide variety of recent samples should be used to cover all important malware types
  - Pay attention to other relevance criteria in sample selection, depending on test requirements
  - Set up a simulated network or use restricted internet, emulating/allowing the most important protocols (but prevent harm)
  - Execute the malware samples in at least two different ways (direct execution from hard disk and execution with an typical infection vector)

Implementation of Test (II)

- Options:
  - Limit the test to special malware categories
  - Use samples of different ages over a longer time span
  - Consider different relevance criteria (localized attacks)
  - Experiment with simulated network settings and network restrictions (always deliver clean or always infected files when malware requests download; respond “good” or “bad” to checks on whether it’s a simulated network or not; restrict different protocols)
  - Execute the malware either in many more different ways or in one special way (as it appears In the Wild, if known)

Some Real Testing Experiences

- Three kinds of products
  - Static detection only, no proactive or generic system guards at all
  - System guards included, but information is not linked together holistically
  - Behavior-based components
- Signature-based detection
  - Even if original sample is not detected via signatures, components downloaded or subsequently created were sometimes detected via signatures
  - Sometimes, behavior-based and signature-based detection would be in place at the same time
- Problem of changing malware behavior:
  - Rarely the case that behavior actually changes
  - The only thing that regularly happened was that malware suddenly stopped working, but would maybe work later again

Questions & Answers

Thank you very much for your attention!

Are there any questions?

Note: Many testing papers can be found at:
http://www.av-test.org → Publications → Papers