Whole-Product Testing Guidelines

Notice and Disclaimer of Liability Concerning the Use of AMTSO Documents

This document is published with the understanding that AMTSO members are supplying this information for general educational purposes only. No professional engineering or any other professional services or advice is being offered hereby. Therefore, you must use your own skill and judgment when reviewing this document and not solely rely on the information provided herein.

AMTSO believes that the information in this document is accurate as of the date of publication although it has not verified its accuracy or determined if there are any errors. Further, such information is subject to change without notice and AMTSO is under no obligation to provide any updates or corrections.

You understand and agree that this document is provided to you exclusively on an as-is basis without any representations or warranties of any kind whether express, implied or statutory. Without limiting the foregoing, AMTSO expressly disclaims all warranties of merchantability, non-infringement, continuous operation, completeness, quality, accuracy and fitness for a particular purpose.

In no event shall AMTSO be liable for any damages or losses of any kind (including, without limitation, any lost profits, lost data or business interruption) arising directly or indirectly out of any use of this document including, without limitation, any direct, indirect, special, incidental, consequential, exemplary and punitive damages regardless of whether any person or entity was advised of the possibility of such damages.

This document is protected by AMTSO's intellectual property rights and may be additionally protected by the intellectual property rights of others.

Whole-Product Testing Guidelines (Protection)

Introduction

This document discusses the issues involved in whole product testing. The document outlines additional issues involved in best practice testing of such products, above and beyond other AMTSO guidelines and best practices. This document is not a comprehensive listing of all such issues.

Unless otherwise defined herein, all terms included in this document are used with their common meaning. The following document should be read in conjunction with AMTSO's *Fundamental Principles of Testing* and other information available at www.amtso.org.

AMTSO documents are best read in conjunction with the other documents on the AMTSO documents page, including *Fundamental Principles of Testing*, *Best Practices for Testing In-the-Cloud Security Products* and *Best Practices for Dynamic Testing*.

Testing of detection rates alone has been a primary focus of many anti-malware tests. This is not unreasonable: after all, threat detection and/or blocking is a core function of most content security products within AMTSO's sphere of interest. However, given the large number of new and dynamic threats facing end users, security companies have developed additional detection and protection capabilities to supplement traditional on-demand detection methods. To provide a realistic picture of today's security products, testers must expand their testing approach to include all relevant product capabilities and subject these products to a test set that correlates with the real threats facing users.

This paper will focus on a balanced look at product effectiveness including detection (and false positive testing against non-malicious programs) in realistic test scenarios that simulate the experiences of real users.

This paper will not focus on usability and performance (as described in AMTSO's *Performance Testing Guidelines*) although such tests could rely on the same best practices.

Anti-malware products are sophisticated applications, and testing a security product in a way that fully and accurately evaluates its capabilities is a challenging process, requiring testing methodology that allows for the assessment of multiple technologies and modules which are integrated into a single package so as to protect a user against malware. Endpoint security products are simply different than they used to be – they have blended countermeasures, which work together to protect the user.

For whole product testing, given its additional complexity, it's especially important for the tester to be clear on what kinds of samples were tested, how products were setup and interacted with, and how effectiveness was measured.

What Constitutes Whole-Product Testing

Testers might consider two methods to evaluate the whole product:

- Whole-Product Testing
- Sum-of-the-Parts Testing

AMTSO recommends Whole-Product Testing (rather than Sum-of-the-Parts Testing) as being the most representative of real world effectiveness.

Whole-Product Testing best mimics a product's use in the real world. Testers are strongly recommended to use the most common (e.g. default) settings and then subject the product to a statistically relevant number of real-world threats. The product is evaluated on how well the user is protected. The product is treated as a whole rather than a sum of its parts. The primary focus is on whether the user is protected against the threat, but it's also instructive for the tester to ascertain which specific capability or capabilities were used to protect the user. The ultimate effectiveness is judged by looking at the threat's impact on the system with and without the product.

Sum-of-the-Parts Testing tries isolating a specific product capability e.g. on-demand scanning or URL filtering, a tester needn't guess on how a product blocked a threat. By combining the results from multiple test parts, a tester can form some conclusions about the product's full capabilities but they would be subjective due to the assumptions made by the tester. The drawback of this approach is that product capabilities often work together to stop a given threat and this interaction cannot be shown through sum-of-the-parts testing. Another disadvantage is that even if one of the product parts was successful in stopping part of the threat as measured by the tester, other parts of the threat may have bypassed the protection and infected the computer (this can be addressed by monitoring or passively assessing the impact on the system).

Facets of Whole-Product Testing

Testers should consider the following when conducting a full product test:

- Stating the Test Purpose
- Selecting Samples
- Setting up Tests and Products
- Introducing Samples
- Handling User Interaction
- Capturing Test Results
- Interpreting Test Results

Stating the Test Purpose

Due to the multitude of test's objectives it is highly recommended that the purpose of the test is stated (settled down) before the next stages are taken in planning the test. As an example, a product test is done in order to check products abilities to protect against "all existing", "newest" or "determined" type of threats. In practice the purpose of the tests is such a matter of course. The results are mostly interpreted as being the ultimate judgment for the quality of product against all existing and not yet existing threats.

Setting Up Environments and Products

The underlying test platforms, components and features chosen, and settings used will depend on the test objective, but in all cases, the tester should strive for a level playing field. The specific test platform, components, features used and settings chosen should be published in the final report or appendix material.

First, the tester must choose the test environment (hardware, operating system, browser version, patch level, applications, etc.) to be used. These choices will most certainly impact the workings of the product and ultimately the results.

Then, once the selection of components is made, choices for product configuration include:

- Default (or the most common) configuration (perhaps appropriate if testing real consumer experience) – note that some products have different default settings based on underlying hardware.
- Tester exercises their judgment in choosing enabled features and settings used (e.g. enable all countermeasures configured for maximum protection, use a vendor-recommended configuration, or choose a different configuration that might offer less protection but may provide a more convenient user experience). This approach may be more applicable for managed environments (like corporate products).
- Testers may consult the vendors to determine the appropriate configuration for a specific Whole-Product test.

The choice of features and settings may also have a significant impact on false positives and performance results. The tester should use the same features and settings for detection, false positive, and performance tests to provide readers a realistic view of what they can expect. In any case the methodology applied (chosen user action chain, enabled product features) requires proper and detailed documentation. This would avoid misunderstandings and complaints regarding the objectivity of a test.

Bare-Metal vs. Virtualized Environment

Virtualization is one particularly important consideration in the platform choice. Some products may behave differently in a virtualized environment. Also, whole product testing includes the execution of the malware, and some malware may behave differently in a virtualized environment.

If the goal is to test effectiveness in a bare-metal environment, testers are encouraged to consider testing on bare-metal test machines. However, virtual environments are increasingly important also and so testers must make a thoughtful choice.

For example:

- It may terminate
- It may remain in memory but inactive
- It may continue to execute, but exhibit only behavior that won't trip detection by behavior analysis (whether static or dynamic)
- It may try to exploit any vulnerabilities in the virtualized environment, for instance to perform some malicious action on the host machine, not just in the virtual environment

It's important to understand both the differences introduced by the virtual environment and the individual behavior of the chosen samples in a virtual environment.

Selecting Samples

A tester should evaluate products using valid, realistic and relevant threats from the field as this is a good indication of whether a product's capabilities are indeed relevant to and effective against the threats facing real users. With whole-product testing, a tester must consider diversity not only in types of attacks and families of malware but also the source of the attacks as all of these factors can impact a product's ability to stop a threat.

As always, a balance between malicious and clean test samples/scenarios must be maintained.

Introducing Samples

The samples should be introduced via their natural propagation vectors (e.g., threats coming via Email should be introduced as Emails). Two approaches could be used:

- Create a testing environment based on "replay" (that helps reproducibility and repeatability of the tests). Live Internet connection of the products should not be altered nor interfered with.
- Use live attacks (this is usually not repeatable but still can be verifiable provided enough logs and records are kept).

In any case care should be taken to ensure fairness with respect to the timing of tests of individual products.

Entry Vectors

The list below provides a number of common entry vectors:

- The user voluntarily, possibly through a social engineering attack, downloads and potentially executes a file from the Web

- The user involuntarily downloads a threat (drive-by download) from the Web
- The user opens a real e-mail, clicks on a link, and downloads and potentially executes a file
- The user opens an e-mail and opens and potentially executes an attachment
- The user receives an instant message and either clicks a link, opens an attachment, and potentially executes a file
- The user accesses a file from a USB stick or other removable media either voluntarily or via auto-run
- The user accesses voluntarily a file from a network share
- The user's computer is infected with a file from the network involuntarily (worms)
- The user receives and potentially executes a file from a P2P network, newsgroup, or other application.

The selection and distribution of vectors should reflect the stated purpose of the test.

Handling User Interaction

In handling user interaction, testers should follow the guidelines laid out in the "Popups and User Interactions" section in AMTSO's *Best Practices for Dynamic Testing*.

The following user interaction models are possible to describe how a user will interact when a product presents a dialog box or other interface components. It's important to note that the user interactions will depend on choices made during product setup or by policies deployed to the product prior to testing:

- No Action – the user takes no action
- Default – simulate by pressing Enter
- Safest action – block everything (could lead to more false positives)
- Worst action – allow everything
- Random action – if this is chosen, tester must include a statistically significant number of test cases
- Predefined action (e.g. always use the top button or the left button) – if this is chosen the tester must document the reasoning behind their choices
- Recommended action (may not be the Default one) – if this is chosen the tester must document the reasoning behind their choices
- "Survey-based" action – if this is chosen the tester must document the reasoning behind the choices.

Note that product may present both modal and non-modal windows and both must be addressed.

Testing all the options is the ideal case but practical considerations will typically necessitate choosing a user interaction model.

The choice of user interaction will depend on the focus of the testing and who the tester envisions as the target user. For enterprise environments, the tester may lean towards more restrictive approaches reflecting the choices an IT administrator might make, and the end-user experience might be more silent given choices already made by an administrator. Consumer-focused whole product testing might suggest a more permissive user interaction model with more dialogs but may not. Although random actions may be realistic for complex human beings, it is not suggested because it may not allow reproducible and credible results.

No matter which user interaction model is chosen, it must be applied consistently to all products tested.

For a given test, testers might also wish to count the number of times the user is asked to make a decision. This data can then be used along with effectiveness results to draw conclusions on the full user experience. It's generally agreed that effectiveness being equal, fewer user interactions is preferred (no interactions being the best).

Capturing Test Results

First and foremost, testers should have their own verification system rather than relying only on vendor product logs (it is recommended that products generate reliable and usable logs). Testers should keep in mind that product logs/reports can be incomplete or incorrect. The tester can use monitoring tools (including network) or passive techniques that compare a system's state before and after exposure to the malware. Logs can be used for anecdotal information perhaps to determine how a product blocked a threat.

In practical terms, it's easiest to look for changes in the system rather than looking for whether the malware is still active, which can be more difficult to determine.

Note: Sometimes it may be necessary to reboot to ensure execution.

Interpreting Test Results

Success can be defined in different ways. Success may be defined as the security product rendering a malware sample unable to execute. Or success might require that the system is brought back to its pre-infection state. Testers must define success clearly for a test and apply the same criteria to all products.

Interpreting test results related to detection and/or blocking can be complex due to the difficulty in determining whether the malware was still able to cause harm to the system or the user after the countermeasures have been applied.

In most cases, testers can treat products as "black boxes" in that it doesn't matter what product capability was successful in blocking a threat, only that the malicious action of the threat was blocked. To provide additional granularity to the test results, testers can attempt to gain deeper insight into how a product blocked a threat. This can be done by accessing product logs or, in some cases, by building additional detection systems.

Counting the number of detections and misses just once on a local system may not reflect all the product's protection capabilities. For example, cloud-based security products use external knowledge and may well be capable of blocking all but the very first few sightings of any specific threat. For AMTSO guidelines on testing cloud-based products please refer to AMTSO's *Best Practices for Testing In-the-Cloud Security Products*. Another example could be a product that would gain the protection knowledge internally, for example by applying artificial intelligence methods. Similarly, a product may lose detections over time. For these reasons it is strongly recommended to track the detection over a period of time, sufficient to cover the entire time span of the attack. The testers should also be aware of the fact that testing cloud-based products against low prevalence threats may trigger a security response.

In all cases, testers should document test methodology, product settings, and other information necessary to interpret the results. Focus on the principles and guidelines as published by AMTSO at www.amtso.org/documents.

User Impact

Finally, testers may attempt to assess the damage caused by the sample to the system (e.g., a keylogger sent private data back to a server). In some of these cases the effects cannot be undone even if the malware is eventually cleaned up. Testers may also give consideration to what happens with user data (like bank account data, email or other personal data).

This document was adopted by AMTSO on May 25, 2010