Guidelines on Facilitating Testability

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.

Guidelines on Facilitating Testability

Introduction

This document covers ways in which testers and vendors can collaborate and share information in order to make testing more efficient and accurate, and to enable external verification of results. The document outlines additional issues involved in best practice testing, 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.

AMTSO documents are best read in conjunction with the *Fundamental Principles of Testing* and other documents on the AMTSO documents page at www.amtso.org.

Logging

More detailed and accessible logging helps both testers and vendors. Although some types of test will need to minimize all interaction with vendors, and run products exactly as normal users would, for larger-scale automated testing it is often a requirement to have access to fine detail about the activities and observations of solutions under test, and what they report about events taking place on the test machine. In many existing solutions, logging may be incomplete, or provided in a format unsuitable for external processing. In order to have their solutions properly and accurately tested, vendors should ensure their products provide adequate information to fully record all events taking place during a test. In situations where logging systems must remain in a non-standard, encrypted or proprietary format which is not best suited to exporting and parsing, vendors should provide tools to convert log data into a more usable format.

For more sophisticated styles of testing it will often be useful to have considerably more information recorded than is currently kept by most solutions, such as information on all pop-ups and alerts displayed by products, and all responses provided by users to requests for confirmation or decisions. Details of products' internal communications such as cloud lookups will also provide useful information for testers when analysing and interpreting test results. A list of all log entries, prompts and other messages and what they mean would be useful, but it would not normally be necessary to cover every possible language.

Having all this additional information available in a format provided by the solutions under test will also be useful to vendors when investigating issues that emerge during third-party tests. Testers are encouraged to provide vendors taking part in their tests with adequate information to diagnose and, ideally, to rectify any problems reported in tests – for example failure to detect or block attacks – and having adequately detailed internal logging simplifies this process considerably.

Much of the information required by testers, and used by vendors to diagnose problems, may not be needed by real-world users as part of their everyday use of solutions. Vendors may find that including options in their products to enable more detailed logging, or providing testers with external tools to

gather or collate the required data, will facilitate better testing and may even be of use in real-world support diagnostics.

Log Entry Examples

Here are some examples of the sort of event-related content that may be required in logs:

- An event occurred
- Time of event
- A unique event ID or reference
- Event category or description
- Source or originator of the event (file, URL...)
- Threat ID/classification
- Action/s taken [automated action, prompt to user, cloud lookup]
- Time taken between event and response/action

And here are some examples of product related content for logging:

- Initialization time
- Update time/version
- Version information

Vendor/Tester Communications

In order to properly test a solution's full range of capabilities, automation is a vital tool for enabling tests executed on a larger scale and thus potentially more accurate. When developing automation systems, testers need to analyse product behaviour in detail and tune their automation to operate and monitor the solutions properly: for example, by responding to prompts for user interaction or capturing pop-ups and other alerts as evidence of responses to threat events. To minimise the additional labour required in developing and maintaining such automation systems, it may be necessary for vendors and testers to communicate openly both on how solutions operate, and on how tests are being run.

It is particularly important that vendors keep testers informed of any changes to how products operate which may affect the running of ongoing tests. Significant changes would include, but are not limited to, areas such as logging format, the style and position of prompts or pop-ups, default configurations, and system requirements. For example, when a number of products went from pop-ups to toasters while an extensive test was being carried out, the need to re-engineer the test in midstream caused significant disruption.

AMTSO strongly encourages open and timely communications between testers and vendors, particularly on issues which may affect how tests can be run, and the organization exists in part with the express purpose of facilitating such communications.

Automation or Scriptability of Solutions During Testing

To enable testers to run fully-automated tests in as efficient a manner as possible, it would be useful for solution developers to provide options or methods to run their products in fully-automated mode, with no user interaction required. This might involve a configuration option where the product always applies a default action when a user prompt would normally be requested, and testers might find it useful to run tests with various approaches to user response, comparing the protection offered by a solution when a user always chooses the most paranoid option versus the most permissive.

Some products may include full options for automating behaviour as standard, but where such controls are not provided vendors should assist testers by providing them with ways of making the product operate in a an automated way. Exactly how this kind of functionality can best be provided will vary from solution to solution, but possible approaches would include command line switches for use in testing only, external configuration files, or templates which can be selected by the tester. It may also be necessary to provide testers with special builds of solutions set to operate in predefined ways.

When using such non-standard products or configurations, testers should be aware of the possibility that products may not accurately reflect real-world behaviour.

Here are some examples of standard user profiles and how they might be applied:

- paranoid user – defaults to blocking or denying any activity queried by their security solution
- "confident" (power-user, super-user) – assumes all activities are as they intended, so allows or whitelists by default
- home user – less well-informed and less interested in spending time deciphering messages, and more interested in getting on with what they were doing; generally permits actions despite alerts, but may sometimes deny if the alert is strongly worded
- corporate user – worried about getting into trouble for breaking corporate policies; generally clicks 'deny', but may occasionally stretch the rules if a warning doesn't sound too serious
- privileged user who may be stretching a system in ways that an unprivileged user wouldn't think of trying, perhaps for evaluation or to test a hypothetical situation – may have a very specific testing objective in mind that requires unusual detail in specific configurations.
- drunken user – selects a random option each time a solution present a request for a decision

Solution Flexibility

While many solution developers may prefer to keep their solutions as simple as possible, to lessen the risk of confusing non-expert users, many types of testing may require considerably more flexibility and control in products in order to produce quality test results without excessive effort.

Some examples of areas which may not usually be configurable in some types of solution, but which may cause significant problems for testing, might include:

- the size, location and detail of logging, as mentioned above

- the ability to scan non-standard areas, such as network drives
- the option to run scans without applying any cleaning, only logging detection results
- the scale and location of quarantine facilities – if large quantities of samples are being moved to a quarantine folder by a solution during a test , it may be necessary to relocate it somewhere other than the standard or default location. Some testers may also find it useful to have easy access to the contents of quarantine folders, to help diagnose the behaviour of solutions during tests.

Tool Sharing

AMTSO encourages vendors to be open to requests for changes from testers, and to do whatever they can to improve the testability of their solutions.

For example, vendors may already have, or may be able to easily produce, tools which would be useful to testers. These could include automation tools used in QA testing, log processing utilities, and file analysis or system monitoring tools for use in sample validation or classification. Testers may also find command-line versions of malware scanners useful for sample classification purposes. Vendors are encouraged to assist testers wherever they can by making such tools and utilities available.

This document was adopted by AMTSO on May 4, 2011

Appendix

A Proposed XML Schema For Log Files

```
productinfo
      company information
      product-specific information
      version
      signature-version (or versions, where multiple engines used)
      updates(s) information: last applied, last asked [true/false
attribute?]

environment (optional: tester should know this anyway)
      OS
      CPU
      RAM

timestamp
      start
      finish

object
      url
      domain
      ip-address
      path&file @md5 (and other hashes)
      process
      registry
      network (object or event? Product-specific?)
      ?items within archive

Classification/ threatID
      malicious-file
      malicious-url
      phishing
      dynamic
      spam
      IM
      Etc. [vendor to classify]

action-taken
      blocked
      quarantined
      disinfected
      deleted
      user-interaction
```

```
action-result
      success
      failed

[Threat category (if not specified above)]
```

XML draft

```
<product>
       <Name>...</Name>
       <Update>...</Update>
       <OS>...</OS>
       <object>
              <name>...</name>
              <type>File/URL/Firewall</type>
              <date>...</date>
              <detection>...</detection>
              <actions>
                     <action>
                            <date>...</date>
                            <name>...</name>
                            <result>...</result>
                     </action>
                   ...
                     ...
                     ...
                     <action>
                     </action>
              </actions>
       </object>
</product>
```