Sunbelt Software

Problems w/ Malware Simulators (and Computer Reports anti-spyware test in particular)

Eric Howes, Director of Malware Research

8/25/2006

The September 2006 issue of Consumer Reports tests anti-spyware applications. In communications with that organization, it has been confirmed that the sole method of testing the antispyware applications in question was the use of Spycar, a set of applications developed by Intelguardians to mimic antispyware behaviour.

There are serious flaws in such a methodology, which I will summarize below.

1. Simulators test against threats that are not real

Strictly speaking, if an anti-malware application allows the action by the simulator, then it is correct in doing so, because the action was performed by a non-malicious program initiated voluntarily and knowingly by the user.

2. Simulators are often based on bad assumptions

All too many "behavioral" or "system change" simulators are based on the assumption that anti-malware applications ought to warn/block on every behavior, system change, or system addition performed by the simulator. In fact, when anti-malware applications do this, they risk inducing a syndrome known as "pop-up fatigue" in users -- a condition where users become inured to pop-up warnings and begin blindly clicking through those warnings. The net effect is to lower system security.

Better anti-malware applications are designed to be "smart" about popping up warnings to users. Those warnings may be conditioned on several things, including whether the "threat" program causing the change is a "known bad" program. Additionally, if the "threat" program is "unknown," more sophisticated anti-malware applications may perform a heuristic check of that "threat" program to determine whether the program exhibits characteristics of a malicious program. Furthermore, some anti-malware applications may condition the display of pop-up warnings on combinations of changes and characteristics, to reduce the risk of pop-up fatigue.

Thus, anti-malware applications may allow a threat simulator to perform a change precisely because the program has determined the simulator and/or the change performed were not likely to constitute a threat. As a result, simple simulators may actually reward poorly designed anti-malware applications for stupidly displaying warnings without exeception (something most users do NOT want), and punish intelligently designed anti-malware applications for displaying warnings only when a likely threat has been detected (which is what most users are actually demanding).

3. Targeting the simulator nullifies the test

If anti-malware applications were to treat the simulator as they do real threats in the wild, then anti-malware scanners would target the program itself as well as key characteristics of the program's behavior. But targeting the simulator nullifies the value of the tests performed by the simulator. The result is an un-real test environment.

4. Simulators impose a false condition/requirement on the tested applications

In real life, the goal of an anti-malware application is to defend a system through whatever means at its disposal, and a protected system is a protected system no matter the means through which it was protected. When testers resort to simulators, however, they impose a new condition on the evaluation of the tested programs: namely, whether or not the program allowed the simulator to execute and complete its tests. Thus, the tester is actually measuring the performance of tested programs on two criteria:

> a) whether the tested program allowed the simulator to execute properly;
>
> b) whether the tested program blocked the changes to the system.

But the first criterion is a false and un-real criterion, because programs that, for whatever reason, fail to allow the simulator to execute and finish its test actions as designed risk being given lower ratings (e.g., a "conditional pass," as in the TechSupportAlert tests -- see: http://www.techsupportalert.com/security_scanners.htm) even when they successfully defended the system.

Put another way, testers risk adapting their testing to accommodate the foibles and limits of the simulator, instead of designing their testing to best reflect the actual threat environment in which anti-malware applications are designed to operate.

5. CR's use of Spycar blatantly disregards the disclaimers of Spycar's authors

CR used Spycar as the sole means of testing anti-spyware programs, yet the authors of Spycar explicitly warn against relying on Spycar for a comprehensive anti-spyware test:

From http://www.spycar.org/Welcome%20to%20Spycar.html:

> "Is Spycar a Comprehensive Test of Anti-Spyware Tools?
> No. Spycar models some behaviors of spyware tools to see if an anti-spyware tool detects and/or blocks it. But, spyware developers are very creative, adding new and clever behaviors all the time. Spycar tests for some of these common behaviors, but not all. Also, with its behavior-based modeling philosophy, Spycar

> does not evaluate the signature base, the user interface, and other vital aspects of an anti-spyware tool. Thus, Spycar alone cannot be used to determine how good or bad an anti-spyware product is. We've used it to find several gaps in anti-spyware product defenses, but Spycar is but one tool for analyzing one set of characteristics of anti-spyware products. A comprehensive review of anti-spyware tools should utilize a whole toolbox, of which Spycar may be one element. Ed Skoudis and Tom Liston wrote an article for Information Security Magazine comparing various enterprise anti-spyware tools, and Spycar was a small subset of our more comprehensive tests. You can see that article here. (http://informationsecurity.techtarget.com/magItem/0,291266,sid42_gci1184258,00.html)"

From http://www.spycar.org/Spycar%20EULA.html:

> "1. DESCRIPTION OF SPYCAR
> Spycar is a suite of tools designed to mimic spyware-like behavior, but in a benign form. Intelguardians created Spycar so anyone could test the behavior-based defenses of an anti-spyware tool. It is intended to be used to see how anti-spyware tools cope with new spyware for which they didn't have a signature. It is not intended to provide perfect anti-spyware tests, or to act as a substitute for any other form of evaluation. In particular, it is designed to test solely the ability of anti-spyware products to conduct behavior-based (non-signature based) detection of spyware."

Insofar as they ignored the disclaimers of the authors of Spycar itself, CR's testers and editors have produced seriously flawed, even misleading tests results. Unfortunately, most readers will have not the slightest clue that anything is amiss with CR's testing.

6. CR's use of Spycar for on-demand scan testing necessarily yields invalid results

Spycar is designed to test the real-time ability of anti-malware applications to block a specific set or actions or system changes that (in theory) resemble those performed by malicious programs. The key assumption or requirement behind these tests is that the anti-malware applications tested are resident in memory and actively monitoring for the types of changes and actions performed by the Spycar test programs.

But CR did not use Spycar to test only the performance of the anti-malware applications when they were resident, running, and actively monitoring for changes. CR also allowed the Spycar programs to make their system changes, after which CR performed on-demand scans with the anti-malware scanners to see if they detected and reversed the changes made by Spycar.

There are several serious problems with this methodology:

a) This use of Spycar falls wholly outside of the uses intended and designed for it by its authors

Spycar was never designed to test on-demand scanning performance or even remediation capabilities; it was designed solely to test real-time behavior detection and blocking.

b) This use of Spycar produces an invalid test bed of "infections"

CR used Spycar to produce a "test bed" of sorts, however, the resulting test bed cannot properly be used to test the actual strength of the anti-malware applications' scan engines or definitions/signatures. The signatures or definitions used by anti-malware scanners for on-demand threat detection are (for good reason) based on specific characteristics of known adware, spyware, and malware programs -- that is by design. The "test bed" produced by CR using Spycar could not be expected to lie within the definitions or signatures of the tested anti-malware applications unless the anti-malware applications had specifically and deliberately targeted Spycar itself -- an action that would effectively undermine and nullify the point of using Spycar at all.

Thus, it is entirely understandable and expected that on-demand tests of anti-malware scanners against the Spycar "test bed" would see many if not most of those anti-malware applications fail to detect and reverse the changes made by Spycar.

Put very simply, CR designed and produced a wholly invalid on-demand test because the test was not designed to test the actual performance of anti-malware scanners against real threats that could reasonably be expected to be included in the signatures of those anti-malware applications.

c) This use of Spycar is premised on a hidden/assumed design requirement

To be sure, there is a class of programs that could be expected to pass CR's on-demand scan with flying colors: change detection and rollback programs. These programs take a snapshot of key system areas, warn those users (after the fact) when those areas are changed, and restore the original settings automatically or on-demand. Whatever the merits and uses of this type of program, such programs cannot be properly regarded as anti-malware applications, as system changes could result from entirely innocent non-malicious activity or circumstances. Such programs are closer to backup and restore programs.

Nonetheless, the on-demand testing performed by CR is essentially premised on the unannounced, un-justified requirement or assumption that anti-malware applications incorporate precisely this type of backup and restore functionality into their suite of protections. Some tested anti-malware applications may have this type of functionality; others may not. Whatever the case, CR's on-demand testing was not an actual test of the anti-malware applications' threat detection and remediation capabilities. It was little for than a check for certain functionality deemed desirable/required by CR's editors, and nowhere was this fact or requirement announced or made clear to CR readers.

d) CR's on-demand testing fails to test critical aspects of the tested applications

CR's use of Spycar to generate a "test bed" of "infections" is flawed not only because those "infections" are not the kind of "infections" that anti-malware applications could reasonably be expected to detect and remove, but because these "infections" do not in fact test the actual detection and remediation capabilities of the tested applications.

This kind of testing says next to nothing about the sophistication of the scan engine, nor does it say anything about the remediation/removal capabilities of the tested applications -- capabilities that are becoming increasingly critical as malware resorts to polymorphic resuscitators, rootkits, and alternate data streams to resist detection and removal.

Still less does it say anything about the coverage and quality of the definitions/signatures of the tested anti-malware applications. Indeed, it would be possible to produce a one-off Anti-Spycar app that passed CR's tests with flying colors but protected against no actual malware in the wild.

The net result of all these failings is that Spycar's on-demand testing of anti-malware applications is simply invalid and meaningless. The tests produced results that say absolutely nothing about the on-demand detection and remediation capabilities of the tested applications.

Eric Howes