MAC HACKING: THE WAY TO BETTER TESTING?

David Harley
ESET, North America
Email david.harley.ic@eset.com

Lysa Myers
Intego, USA
Email lysamyers@gmail.com

ABSTRACT

Anti-malware testing on the *Windows* platform remains highly controversial, even after almost two decades of regular and frequent testing using millions of malware samples. Macs have fewer threats and fewer prior tests on which to base their methodology, so establishing sound mainstream testing is even trickier. But as both Macs and Mac malware increase in prevalence, the importance of testing software intended to supplement the internal security of *OS X* increases too.

What features and scenarios make Mac testing so much trickier? We look at the ways in which *Apple*'s intensive work on enhancing *OS X* security internally, with internal detection of known malware, has actually driven testers back towards the style of static testing from which *Windows* testing has moved on. And in what ways might testing a Mac be easier? What can a tester do to make testing more similar to real-world scenarios, and are there things that should reasonably be done that would make a test less realistic yet more fair and accurate? This paper examines the testing scenarios that are unique to Macs and *OS X*, and offers some possibilities for ways to create a test that is both relevant and fair.

1. The testers dilemma: static testing in a dynamic threatscape.
2. Reconfiguring the environment and staying real-world.
3. Sample management and expectation management.
4. Mac anti-malware testing: the next generation.
5. What this tells us about testing on other platforms.

INTRODUCTION

AMTSO's sometimes stormy history [1], many years after the anti-malware industry's early attempts to offer guidance to the embryonic testing industry [2], indicates how controversial testing on the *Windows* platform can still be. Surely Mac testing, with a tiny estimated total sample population compared to the tens of millions of known *Windows*-targeting samples, must be less contentious, with few threat families and generally lower infection rates?

However, the challenges on testing *Windows* security products are not the same as those on *OS X*, or even *iOS* or *Android*. Each operating system is its own ecosystem, and has its own security challenges and assets. The threats for each platform reflect those, as should the products and tests that evaluate them.

The comparatively small number of threats for *OS X* at first glance seems to be a boon for testers: finding a statistically meaningful number of samples isn't difficult for a tester with a comprehensive, up-to-date collection. Testing with all known malware may be almost as quick – for a static test, at least – as using a smaller percentage of the most prevalent samples or families. But the difficulty in choosing relevant samples remains, for completely different reasons. In recent years, the handfuls of threats affecting the wider base of Mac users have affected increasingly dramatic numbers of users.

In 2012, Flashback pushed the number of infected machines way, way up to 600,000 – 700,000 [3], an impressive number when you consider that while some *Windows* outbreaks seem a lot larger, the *Windows* market share is very much greater [4]. But Flashback was hardly the only threat to affect Mac users. Without Flashback's striking impact, the upward trend is more gradual, but its significance as an indicator should not be dismissed as an outlier. In fact, the percentage increase in the number of threats over the last year or two has been dramatic (something in the order of 100%), but that is in part because the starting figure was so low [5].

Comparison between the prevalence of Conficker and Flashback may be misleading but does represent a gradual convergence between the two platforms in (financial) motivation [6]. From the outset, many profit-motivated threats for *OS X* have been 'multi-platform' or were created by the same gangs that have long been attacking *Windows* users [7].

The most consistent recent growth area in Mac malware has been targeted attacks, not generic, untargeted malware. *OS X* almost seems to attract more (proportionally speaking) in the way of APTs [8, 9], mostly targeting Non-Governmental Organizations (NGOs), presumably for political reasons [10]. Statistics may also be misleading because in some cases, once samples are shared *Apple* embeds some form of countermeasure into the OS itself.

Apple and *Microsoft* demonstrate comparable performance over time on vulnerability patching and proactive defensive technologies, like application sandboxing. These two companies have adopted very different approaches when it comes to informing the general public and the security industry about their actions, though the increasing volume of Mac malware has seen *Apple* move towards a closer but not particularly public relationship with the anti-malware industry. *Apple*'s muted approach to relations with the security industry and its inclusion of its own simplistic anti-malware component in *OS X* has introduced unexpected complexities into the product-testing arena.

PHILOSOPHY AND SECURITY

Even in the 1990s, we didn't really want to see *Apple* produce a Mac anti-virus analogous to *Microsoft*'s ill-fated first attempt at bundling anti-virus with the operating system [11].

Over the lifetime of *OS X*, *Apple* security research has gone beyond denial ('there are no Mac viruses') to cooperating with the anti-malware industry. While most of the layers of *OS X* system security are generic (and directly comparable with similar defensive layers in recent *Windows* versions [12]) *Apple* has included signature detection within the operating system with *XProtect* [13]. While *XProtect* – unlike *Microsoft Essentials*, incorporated in *Windows 8* into *Windows Defender* – cannot be directly compared to a full-blown commercial anti-virus/anti-malware product, it is close enough to raise similar concerns.

> 'I am concerned that Apple may not take the threat seriously enough to produce and maintain a consistently effective defence: while you can argue that any defence is better than none, … in the long run … mediocre protection would do more harm than good … because Apple's customer-base will tend to overestimate the effectiveness of any measure Apple do take, the same way that they already overestimate the value of the free anti-malware tools already available.' [11]

XProtect functionality is far narrower than the optional MSE engine or an independent anti-malware solution with true on-access capability, though some of that functionality is also present in a fully up-to-date *OS X* system with full security enabled [12, 13]. Critically, the range of threats it detects is limited, as is the range of vectors it covers. Yet it goes further in its integration with the OS than *Microsoft*'s solution. The use and choice of an anti-virus package or security suite is voluntary up to the point of installation, and even then it can be configured in order to reduce its impact on the system. *XProtect* and *Gatekeeper* are not immovable, but their integration into the system is harder to resist (and we're not saying that the everyday user *should* resist), they impact directly on the functioning of third-party security software, and they've dramatically modified – and compromised – the approaches available for *OS X* product testing.

From a testing perspective, there are two main components within *OS X* that need to be considered: *XProtect* and *Gatekeeper*. Each takes a slightly different approach to preventing malicious files from executing, and the effects will differ depending on what level of protection the user has elected to enable. *XProtect* is comprised of two files: XProtect.plist and XProtect.meta.plist.

XProtect.plist

XProtect.plist is intended to help detect malware purely reactively [14]. The detections are quite specific, far from comprehensive, and there is no heuristic or generic detection to help detect new malware or variants. However, it effectively invalidates dynamic testing on systems where the sample is known and detection has been added to XProtect.plist.

XProtect.meta.plist

This component has recently been used to take a higher-level approach to preventing potential malware attacks, primarily by preventing older browser plug-ins (namely Java and Flash) from working. In emergency, zero-day situations, *Apple* has used this to completely disable plug-in components.

Gatekeeper

Gatekeeper is *OS X*'s answer to the walled garden of *iOS*, allowing users to reject apps to different, specific levels based on whether *Apple*-approved developers have signed the code, or whether *Apple* approved the app itself.

It has three basic settings [13]:

- install and run only apps from the Mac App Store
- also install and run apps that have a Developer ID
- install and run apps from anywhere.

Control-clicking allows the user to override his default setting, so as with Alan Solomon's perfect anti-virus [15], the decision on whether to install remains with the user.

Gatekeeper can be helpful when a threat relies solely on social engineering to get you to infect your system but if the malicious app is signed and appears legitimate, or exploits a vulnerability not (yet) patched in order to install silently, it is ineffective.

Both *Gatekeeper* and *XProtect* will ignore a file copied from fixed media or via applications that aren't on its select list of applications to monitor, or if it's one of the file-types *OS X* considers safe. And neither monitors files on egress.

One consumer organization's test is notable for (in our opinion) inappropriate and ill-conceived advocacy for the (*Windows 8*) operating system as a complete solution to the malware problem, but asks one very pertinent question: '… would you trust security software built into your computer's operating system?' Well, for many people – including many *Apple* users – the answer is clearly yes, but should they? At any rate, should that trust be absolute? [16, 17]

Either way, these utilities have a significant impact on how testing needs to be set up to accurately reflect how a product performs on a real-world Mac.

STATIC TESTING IN A DYNAMIC THREATSCAPE

Comparative testing in the Mac world introduces an extra competitive layer. Products are not only in competition with each other, but with *Apple*, in that dynamic or on-access testing is *only* practical with samples for which *Apple* hasn't implemented signature detection yet *or* with samples that *XProtect* sigs won't catch in a real-life infection scenario.

Gatekeeper can be overridden manually, though that might still be a problem during an intensive test unless the *Gatekeeper* response is automated, or the utility is disabled completely [18].

The limitations of the signature-based XProtect.plist utility mean that it can be 'evaded'. Certainly if you're able to test before an *XProtect* signature is added, it's unlikely that the utility will interfere with that testing segment, though static batch testing could still be derailed by the inclusion of samples for which a signature *does* exist.

That window before *XProtect* covers a new threat discovered by *Apple*'s own or other researchers can be days or weeks wide, leaving machines that are unprotected by mainstream anti-malware exposed to potential infection. Not all threats detected by anti-malware ever get added to *XProtect*: malware that is less 'dangerous' or prevalent never makes it into *XProtect*, which is unfortunate if you're one of the few victims. (Bear in mind that many recent Mac threats are highly targeted, aimed at activists among particular ethnic groups and NGOs.)

The window of opportunity that is available to the malware is also available to the tester. But that window closes when an *Xprotect* signature has been added. At this time, Mac testers are almost entirely focused on retrospective testing with samples that are already 'XProtected' [19] or at any rate assumed to be [20, 21, 22].

If a tester does need to work with retrospective 'snapshot' testing [23] of old or 'extinct' samples – and the paucity of Mac-specific threats makes it impractical to do a snapshot test that isn't retrospective and doesn't use 'stale' samples – the question arises as to whether there is a meaningful

distinction between 'stale' and extinct [24]. There are presumably still systems carrying system-targeting malware from OS9.* and earlier, and it's probable that all commercial anti-malware detects it, but it's unlikely that any tester uses it. (Yet we still see *Windows*-oriented tests that include antique DOS viruses: indeed, *ESET* telemetry indicates that a steady trickle of boot sector viruses is still traversing the threatscape, many years after *Windows 95* is supposed to have killed them all off [25]). Conficker-infected machines continue to trip alarms even though the Conficker botnet itself is essentially gutted, and some marketing-oriented sources suggest that the number of machines still infected with Flashback is surprisingly high. If so, it may be due to people running outdated versions of the operating system. In *Apple*'s case, this could be because of hardware that isn't supported by recent OS versions.

The tester with a preponderance of stale samples can simply disable *XProtect*, much as Flashback did [26]. But is this sufficient?

Disabling system-integrated protection (including both *Gatekeeper* and *XProtect*) moves you away from the 'real world', at least as most people experience it. Not only is the system 'untypical' of real-life user experience, but disabling one aspect of the inbuilt security may break something else. If you're testing in a live network scenario, you may have just introduced a risk to other vulnerable systems [27, 28].

Dynamic testing on an unpatched, de-XProtected system or pre-*XProtect* OS is perhaps real-world in the very limited sense that unpatched and un-updated systems do undeniably exist in the real world, though the number of up-to-date scanners that will run pre-*Snow Leopard* (*OS X 10.6*) has already decreased dramatically. And something feels very wrong about using obsolescent OS system versions (unless you're doing *Virus Bulletin*/*ICSA Labs*/certification-test-style platform-specific tests) in order to test an additional layer of security that can't be tested on a *current* OS version. And 'breaking' a current operating system (or, indeed, an app under test) in order to isolate a single layer of tested functionality is a long way from the principles of whole-product testing and *isn't* representative of the average customer's real-world experience.

But how else do you ascertain whether a product's specific detection of sample X differs between on access and on demand?

Could testers grab a test image from the first day of the time period they intend to test, so that any malware that came out between that time and the first day of testing would be tested as if it's the time period before *XProtect* updating? There is usually a several-day to several-week lag in updating against even the most prevalent malware. This would be very much like the freezing of test sets and product versions/updates formerly much used [29]. If it seems a bit retro, it nevertheless represents a step or two forward from static testing. We might even see a return for Time to Update (TtU) testing, though there are some of us who would hate to see it [30]. However, sound real-time testing could address this acceptably. As each threat is acquired, it's run against the products under test. At the end of the test period, the individual results can be compiled into the final report. The relatively infrequent appearance of new *OS X*-targeting threats makes this scenario much more feasible and infinitely less resource-intensive than taking the same approach for *Windows* testing, where tens of thousands of new, unique malicious binaries are processed each day.

ON DEMAND VERSUS ON ACCESS

How many *Windows* users nowadays routinely run an on-demand scan (i.e. unless they have particular reason to believe that their machines have been infected)? More than the number of people who still run sheepdip systems, perhaps, but far fewer than the total number of people who run security software. Perhaps because fewer pundits advise them to, or because they prefer the minor performance hit that comes when a file is scanned as it's opened to the greater hit of a whole-disk scan (even if it's a background scan). Even if they're not particularly familiar of the meaning of terms like sandboxing, emulation, behaviour analysis and so on as they're used in anti-malware technology, they might nonetheless be aware that some scanners are better at detecting some threats on execution than they are at detecting them by passive scanning.

In the Mac arena, the opposite is true, at least in the perception of some testers. As one of the authors wrote about a review in Thomas Reed's Tech Corner [23]:

> '[Reed] states that he is testing without the on-access scanner, which [he believes] is how detection would happen in most real-world situations.'

Is this really the case today? Frankly, we doubt that users of commercial Mac anti-malware do heavy on-demand scanning nowadays except in circumstances like these:

- In some corporate environments where security staff have an old-school view of the necessity and desirability of running a regularly scheduled on-demand scan. We suspect that the same is likely to be true of some *Windows*-using corporates.
- In some environments where a not-altogether-commercial grade on-demand scanner is still considered to be sufficient, either as an email filter or for scanning files on disk via a third-party app.

In fact, the use of static testing of on-demand components rather than whole product testing remains a common scenario amongst testing labs, sometimes because static testing is cheaper and, in principle, simpler to implement [31].

In many cases, though, modern scanners use emulation in on-demand scanning so that a program being scanned is allowed to execute in a virtualized or emulated environment. Nonetheless, testing that *assumes* that ability in all contexts is not maintaining a level playing field. And in the *Windows* environment, it's getting hard to justify something that falls so short of whole-product testing: the all-folder on-demand or scheduled scan is the exception rather than the rule. If it persists in some commercial-grade Mac security products, that may be because malicious Mac-directed programs have less need to be technically complex than their *Windows*-directed siblings, and that may be reflected in comparatively laid-back anti-malware technology. It's not surprising if vendors don't use resource-intensive technologies that are not – or not yet – needed in the context of *OS X*.

Mac product testing, however, is largely based at present on the (increasingly inaccurate) assumption that the difficulties of on-access scanning need not be addressed since Mac malware is less likely to be self-protected by the kind of

anti-forensic obfuscation that characterizes so much *Windows* malware.

RECONFIGURING THE REAL WORLD

Can a test run on a fully patched system running the latest *OS X* versions be on-access – i.e. where potentially malicious objects are scanned when a file is accessed (opened, executed, copied and so on)? Only if the samples aren't detectable by active system utilities, because otherwise the OS won't allow malware to execute. Testers don't necessarily have time to spend wrestling with *OS X* internals or have the resources to (a) acquire, validate and test with samples before the XProtect.plist window closes, and (b) test longitudinally (i.e. over time) so as to accommodate that window of opportunity. Testers that do this routinely tend to be certification testers who have the capacity to run longitudinal testing because that's essentially what their vendor customers are paying for.

Admittedly, there are ways to 'de-patch' the relevant components of the OS, but is that real-world testing? If the OS is able to intervene because the malware is a variant that it recognizes, that's real-world in a sense, but it's not whole-product testing. At a time when mainstream testers are anxious to implement whole-product testing in accordance with AMTSO [32] guidelines, it seems that testers simply aren't able to do so on *OS X*, whether for technical reasons or because of resource issues. (There are analogous issues on other platforms, especially mobile devices.) And that's OK as long as it's clear to readers of test reviews that what they're looking at is a compromise, not a perfect reflection of a product's capabilities in the real world. Not only because it's not a guide to its capability regarding malware that isn't already neutralized, but because static testing isn't conclusive proof of the detection that it would offer (or would have offered) in the absence of the operating system's own defences.

It has been suggested that there's less differentiation between on-demand and on-access scanning results in a Mac test because Mac products don't use behaviour analysis. There is, in fact some truth in this: many Mac scanners do make less use of advanced proactive detection techniques than commercial-grade *Windows* scanners, even when they come from a company that has a *Windows* product range. This is especially true in the context of Mac-specific threats, and in fact some products are little more than a Mac-friendly shell around a ported *Windows* or *Linux* engine with enhanced awareness of Mac and cross-platform threats, but less sophistication in the range of detection technologies employed. However, it's an oversimplification to imply that any but the most basic scanners make *no* use of behaviour analysis.

MAC ANTI-MALWARE TESTING: THE NEXT GENERATION

Mac anti-malware is more critical in terms of early detection (pre-*XProtect*), at least for malware that is widespread enough to rate an *XProtect* signature. It's even more critical if you're a victim of malware that never gets an *XProtect* signature: so maybe we should talk about criticality in terms of (a) early detection, (b) low-prevalence malware, and (c) removal/disinfection – there is no removal function in *XProtect* except for malware that *Apple* perceives as an outbreak-level threat.

In other words, in the Mac arena, the whole concept of a snapshot test is flawed: a better route is to test longitudinally (response-based, ongoing, real-time). Maybe *all* detection testing should be longitudinal, but it may be easier to get away with batch testing over a few days with *Windows* products.

However, longitudinal testing is not so much a test of raw detection as it is a test of responsiveness and the product's relationship with the other players – cooperation with other vendors, with *Apple*, other security sectors, external agencies... That's not a bad thing, but it's not particularly tester-friendly because it pushes them towards expensive time- and resource-intensive whole-product testing methodologies that their clients may not want to pay for.

In fact, good relationships at that level actually reduce the differences between vendors, whereas testers are often obliged to exaggerate the differences in order to generate a ranking – more often than not, in order to meet the demands of the magazines, consumer organizations and so on who are frequently their customers, so testers would have to focus more on other aspects (whole-product testing) to restore differentiation.

LIES, DAMN LIES, STATISTICS AND TESTING

Statistical relevance is a highly sensitive topic in testing [33]: even if we sidestep the Sisyphean task of raising the general ability to interpret – let alone generate – test results correctly. One of the less daunting aspects of the Mac testing problem is the relatively small size of the total Mac-specific sample population: how many years will it take to generate the same number of unique malicious samples as we currently see for *Windows* in a single day? Thus, a number of samples that would seem ridiculously small for a *Windows* test (or pseudo-test [34]) could be considered quite comprehensive in the Mac context [23]. However, it can certainly also be argued that the manageability of the sample pool means that we could reasonably expect a very high proportion of the potential population to be used in a test – though this may vary according to test objective and the functionality under test – and still be affordable.

It's only fair and responsible to make it clear what approximate proportion of the presumed total threat population your sample set represents, irrespective of platform. Statistical validity – in the limited sense of the sample set being representative of the total population is more achievable as long as testers can acquire sufficient samples in a timely manner. We suspect, though, that even some mainstream testers may not be as well connected in Mac sample acquisition sharing as they are in the *Windows* arena.

Clementi [35] points out that prevalence can still play a part even though the total volume of Mac malware remains low. He cites the hypothetical example of malware which is highly targeted – and therefore of very low prevalence – that may or may not be shared with other vendors but falls into the hands of testers. Certainly there are testers who encourage vendors to provide them with samples other companies are unlikely to detect, probably in the hope of establishing a clear ranking. Is that a fair approach, or is it allowing vendors with an 'all's fair in love and testing' ethic to game the test? We believe that better cooperation between vendors and testers is a better way

to address the issue of prevalence than encouraging artificial inflation of scores by cherry-picking samples.

A further difficulty arises with sample acquisition and sharing. While applications and many other kinds of Mac program look like a single entity represented by a single icon (e.g. Preview.app), examination with 'Show Package Contents' generally shows the 'bundle' to contain a directory that may contain a wide variety of files, subfolders, resources and so on. Where a malicious application takes the form of an .app bundle – it could in theory be one of many bundle types – vendor detection may be limited to unequivocally malicious bundle components. However, if the full bundle is not available to the tester, the malware cannot be fully installed, making on-access detection impractical.

A related point holds for *Windows*, especially for web-based malware. Vendors have rightly queried test results that include malware components that are not of themselves malicious in other contexts. Communication between the vendors and testers continues to be very important, as not all parts of malware can – or should – be detected in all cases.

TESTING ON OTHER PLATFORMS

These issues invite speculation about what we might or should expect from third-party security software in an environment that tries to plug its own gaps non-generically, and what we might expect from testers.

Of course, any responsible vendor attempts to reduce the operating system attack surface generically. If the future of personal computing is the mobile device – it probably isn't, at least not exclusively, but that's another debate – it's not surprising that recent mobile operating systems attempt to build in strong generic protection, learning perhaps from the mistakes of early generations of operating system – not only desktop systems, but mobile. (Think *Symbian* [36].)

Apple's own *iOS* includes an 'iron hand' approach to app-sandboxing that effectively precludes the use of full-strength malware-detection software, though on-demand scanners for *iOS* do exist [37] and there are a number of apps that offer some form of web filtering (malicious or child-unfriendly URLs, QR code reading, and so on). It also imposes a layer of App Store whitelisting that has so far kept *iGadgets* an almost entirely malware/anti-malware-free zone, though malware-free is not the same as totally secure [38]. However, virtually all malware native to the platform relies on jailbreaking [39], so that an approved on-demand scanner is likeliest to detect:

- malware that isn't native to *iOS* but might use the *iGadget* as a gateway to vulnerable systems (heterogeneous malware transmission)
- borderline apps that are closer to the 'possibly unwanted' class than to unequivocal malware
- *iOS*-specific malware that can only take hold on a jailbroken device.

Apple acts quickly to fix vulnerabilities that facilitate jailbreaking, but the number of devices that are or have been jailbroken probably runs into millions, according to *Cydia et al.* Nevertheless this promptitude of action has certainly contributed to the scarcity of vulnerabilities that has made the process by which jailbreak teams chain together those vulnerabilities more difficult. The work needed to create malware on *iOS 6* is far, far greater than it would have been on earlier versions. The combination of the tireless efforts of jailbreakers and *Apple*'s striving to keep its walled garden has been rather beneficial to the health of the *iOS* ecosystem.

It has been suggested that security software might itself be made available for jailbroken devices [40] with enhanced functionality (posing both practical and ethical issues in terms of *Apple*'s own opposition to relaxing its grip on the marketplace), and it's hard to see how conventional detection on an approved device could be tested without creating a thoroughly artificial scenario.

Although *Windows 8* RT was surprisingly quickly targeted for what amounts to a tethered jailbreak [41], it uses a similar model to *Apple*'s: only *Microsoft*-approved software (a handful of *Microsoft*'s own desktop applications and *Windows 8*-specific apps from the *Windows Store*) can execute on an unsubverted machine. The operating system includes *Windows Defender* and Smartscreen, and commercial RT-specific anti-malware has several times the scarcity value of hen's teeth.

For both these platforms, therefore, it's hard to see commercial advantage right now to serious anti-malware testing in any form.

Android, however, is a different kettle of phish (and more to the point, trojans). *Google* has damaged its credibility somewhat, attempting to divert customer attention from any suggestion of risk to its customers by lambasting the 'scammers and charlatans' of the mainstream security industry [42, 43, 44]. In the real world, however, *Android* has been building up – or down – its reputation as (in terms of vulnerability to malware) 'the new *Windows*' (not to mention the new *Symbian*) [45, 46]. A recent report [47] indicates that out of 149 recent mobile threats (families and variants), 136 are specific to *Android* (the rest being specific to the declining *Symbian* platform). *Google* also has mechanisms for checking/approving apps submitted to its own *Google Play* store and is capable of disabling download of malicious apps retrospectively and remotely. However, it doesn't prevent *Android* users downloading apps from unregulated repositories, and indeed has the unenviable distinction of having carried hacked apps in its own store [48].

Consequently, *Android* is considered the malware- and anti-malware-richest mobile environment. There is a plethora of both free and for-fee scanning apps from mainstream security vendors as well as less-recognized sources, and a sufficiency of samples is available for realistic on-demand and on-installation testing [49]. Furthermore, tests already carried out suggest a wide range of capability between products (contrast between free and mainstream detection scores is particularly striking [50]), suggesting a very real need for continued testing that allows end-users to make an informed evaluation of the options available to them. We note also that *Android* anti-malware is already receiving academic scrutiny – in some instances somewhat tied to a somewhat old-school PC-oriented view of anti-malware technology [51]. Yet this may be a precursor to future changes in threat technology. It may well also be that AMTSO's mobile testing guidelines, currently in process, will be primarily focused on *Android*-focused testing in the first instance.

CONCLUSION

A smaller threat population has potential statistical advantages, but there is little history of dynamic or whole-product testing

for *OS X*. There are other problems, though: *Apple*'s (only partially) successful interventional model, varying levels of experience (both in the operating system and in testing) and inconsistency in *OS-X*-specific threat tracking.

Is *OS X* so well-armoured that the only time you need a modern anti-virus product is in the following scenarios?

- For detecting new malware that is capable of evading *Apple*'s current defences. (Even though Mac scanners tend to be weaker on proactive detection than their *Windows* equivalents, commercial-grade software includes generic detection capabilities generally superior to *XProtect* signatures.)
- For detecting threats that aren't really on *Apple*'s radar. (This includes cross-platform malware, threats targeting platforms like Java that *Apple* doesn't consistently support, heterogeneous transmission of non-*Apple* malware, threats that rely on social engineering rather than sophisticated binaries, and Possibly Unwanted Applications.)
- For detection by signature for the purpose of remediation. (Larry Bridwell suggests [52] that signature detection is mostly for remediation, whereas contemporary anti-malware is more about protection by behaviour analysis, traffic analysis, active heuristics, sandboxing, reputation – or should be...)

If that's the case, testers certainly need to be moving away from static, snapshot testing to a response-based, ongoing, real-time methodology, focused not only on detection but on remediation.

It could also be said that paying for commercial anti-malware is not just protection for an individual machine, but also constitutes a kind of (voluntary) tax you pay for the kind of specialist hard-core analysis of individual samples that other technologies make use of but don't acknowledge. How effective would alternative technologies be if the kind of research and analysis carried out by anti-malware labs stopped happening because no-one will pay for commercial products anymore [53]? How much free but commercial-grade anti-virus would survive if there were no for-fee AV-incorporating security solutions to underwrite the costs of producing free versions?

Apple's relationship with the industry highlights the need to consider this question. There's a degree of communication and exchange of malware-related information between *Apple* and the security industry that would have been unthinkable not so long ago, and *Apple*'s customers are safer for it. But not only is the average customer unaware of that relationship, he believes that *Apple* is protecting him all by itself and with 100% efficacy. And is then taken aback at finding that *Apple* isn't detecting all the samples in a test [20].

It's not necessarily wrong to use static testing, as long as you make very, very sure that your audience is aware of the *limitations* of that approach [19]. Unfortunately, testers aren't always very good at admitting all the limitations of their methodology, just as vendors – or at any rate their marketing departments – are often reluctant to admit the limitations of their products. No wonder it's so difficult for the consumer (individual or corporate) to come to a fully informed buying decision.

We do not have all the answers to resolving this tension, but without such resolution, we risk a breakdown – in the Mac context at least, and probably in a much wider context – of the symbiotic relationship between vendors and testers, and that will hurt both parties. At the very least, the willingness of vendors to expose themselves to testing on *OS X* will be compromised.

ACKNOWLEDGEMENTS

The authors would like to thank friends and colleagues in the security and testing industries for sharing their own thoughts on these issues with us. In particular, Andreas Clementi, Peter Stelzhammer, Larry Bridwell, Aryeh Goretsky and Andrew Lee.

REFERENCES

[1] Harley, D. After AMTSO: a funny thing happened on the way to the forum,.EICAR conference proceedings, 2012. http://smallbluegreenblog.wordpress.com/2012/05/10/after-amtso-a-paper-for-eicar-2012/.

[2] Bontchev, V. Analysis and Maintenance of a Clean Virus Library, 1993. http://www.people.frisk-software.com/~bontchev/papers/virlib.html.

[3] Cluley, G. http://nakedsecurity.sophos.com/2013/04/03/flashback-mac-malware-author/.

[4] Cobb, S. http://www.welivesecurity.com/2013/01/31/straight-facts-about-mac-malware-threats-and-responses/.

[5] ESET. http://www.eset.com/us/mac-malware-myths-and-facts/.

[6] Krebs, B. http://krebsonsecurity.com/2013/04/who-wrote-the-flashback-os-x-worm/.

[7] Cluley, G. http://nakedsecurity.sophos.com/2010/11/24/apple-mac-malware-short-history/.

[8] Harley, D. http://macviruscom.wordpress.com/2013/05/19/mac-spyware/.

[9] Myers, L. http://www.intego.com/mac-security-blog/new-mac-spyware-discovered-osxdockster-a/.

[10] Harley, D. http://www.welivesecurity.com/2012/12/04/spying-on-tibetan-sympathisers-and-activists-double-dockster/.

[11] Harley, D. http://www.welivesecurity.com/2009/08/26/mad-macs-beyond-blunderdome/.

[12] Myers, L. http://www.intego.com/mac-security-blog/do-os-xs-built-in-security-features-offer-good-enough-protection/.

[13] Apple. http://www.apple.com/osx/what-is/security.html.

[14] James, P. http://www.intego.com/mac-security-blog/how-the-anti-malware-function-in-apples-snow-leopard-works/.

[15] Solomon, A. The Perfect Antivirus. http://groups.google.com/group/alt.comp.virus/msg/d3d5bd6bf37004b8?pli=1.

[16] Harley, D. http://antimalwaretesting.wordpress.com/2013/01/25/which-hunting/.

[17] Moreton, J. http://conversation.which.co.uk/technology/windows8-anti-virus-security-software-tops-table-microsoft/.

[18] University of Chicago IT Services. http://answers.uchicago.edu/page.php?id=25481.

[19] Harley, D. http://antimalwaretesting.wordpress.com/2013/01/10/mac-testing-static-versus-dynamic/.

[20] Reed, T. http://www.reedcorner.net/mac-av-detection-rates/.

[21] Reed, T. http://www.reedcorner.net/mac-anti-virus-testing-01-2013/.

[22] Harley, D. http://www.infosecurity-magazine.com/blog/2013/1/29/mac-av-testing-how-useful-is-it/781.aspx.

[23] Myers, L. http://www.intego.com/mac-security-blog/that-anti-virus-test-you-read-might-not-be-accurate-and-heres-why/.

[24] Reed, T. http://www.reedcorner.net/variant-of-smssend-slips-past-xprotect/.

[25] Goretsky, A. Personal communication.

[26] Haslam, K. New Version of Flashback Eludes Apple's XProtect, 2012. http://www.macworld.co.uk/macsoftware/news/?newsid=3353360.

[27] AMTSO Guidelines for testing Network Based Security Products. http://www.amtso.org/amtso---download---guidelines-for-testing-network-based-security-products.html.

[28] AMTSO Best Practices for Testing In-the-Cloud Security Products. http://www.amtso.org/amtso---download---amtso-best-practices-for-testing-in-the-cloud-security.html.

[29] AV-Comparatives. http://www.av-comparatives.org/images/stories/test/ondret/avc_report25.pdf.

[30] Lee, A.; Harley, D. Anti-Malware Evaluation and Testing, in AVIEN Malware Defense Guide for the Enterprise, Ed. Harley, Syngress, 2007.

[31] Harley, D. http://www.infosecurity-magazine.com/blog/2012/10/13/the-test-of-time/668.aspx.

[32] Anti-Malware Testing Standards Organization. http://www.amtso.org/documents.html.

[33] Kosinar, P.; Malcho, J.; Marko, R.; Harley, D. AV Testing Exposed. Virus Bulletin 2010 Conference Proceedings. http://go.eset.com/us/resources/white-papers/Kosinar-etal-VB2010.pdf.

[34] Bloggit, O.M. https://antimalwaretesting.wordpress.com/2013/01/02/journalisms-dirty-little-secret/.

[35] Clementi, A. Personal communication.

[36] F-Secure. http://www.f-secure.com/static/doc/labs_global/Research/Mobile%20Threat%20Report%20Q3%202012.pdf.

[37] Intego. http://www.intego.com/mac-virus-barrier-ios.

[38] Harley, D. http://www.infosecurity-magazine.com/blog/2012/7/20/pickpockets-in-the-app-marketplace/602.aspx.

[39] Wismer, K. http://anti-virus-rants.blogspot.co.uk/2012/02/is-iphone-really-malware-free.html.

[40] Townsend, K. http://www.infosecurity-magazine.com/view/26013/ios-551-jailbreak-done-ios-6-jailbreak-pending/.

[41] Wisniewski, C. http://nakedsecurity.sophos.com/2013/01/08/windows-rt-jailbroken-shows-its-windows-8-roots/.

[42] DiBona, C. https://plus.google.com/u/0/+cdibona/posts/ZqPvFwdDLPv.

[43] Harley, D. http://macviruscom.wordpress.com/2011/11/21/mobile-av-another-charlatan-scammer-hits-back/.

[44] Harley, D. http://macviruscom.wordpress.com/2011/11/21/memoirs-of-a-charlatan-scammer/.

[45] Harley, D. http://macviruscom.wordpress.com/2013/03/11/android-malware-and-blaming-the-victim/.

[46] Cobb, S. http://www.welivesecurity.com/2013/03/11/android-security-issues-does-a-microsoft-windows-analogy-make-sense/.

[47] F-Secure. http://www.f-secure.com/static/doc/labs_global/Research/Mobile_Threat_Report_Q1_2013.pdf.

[48] Yeung, K. http://thenextweb.com/insider/2013/05/26/british-broadcaster-sky-news-has-all-of-its-android-apps-hacked-by-the-syrian-electronic-army/.

[49] AV-Comparatives. http://www.av-comparatives.org/wp-content/uploads/2012/09/avc_mob_201209_en.pdf.

[50] AV-Test: http://www.av-test.org/en/tests/mobile-devices/android/mar-2013/.

[51] Rastogi, V.; Chen, Y.; Jiang, X. Evaluating Android Anti-malware against Transformation Attacks, NorthWestern University, 2013. http://list.cs.northwestern.edu/mobile/droidchameleon_nu_eecs_13_01.pdf.

[52] Bridwell, L. Personal communication.

[53] Harley, D. Anti-Virus: Last Rites, or Rites of Passage? Virus Bulletin, February 2013. http://geekpeninsula.wordpress.com/2013/05/01/virus-bulletin-articles-1/.