NICHOLAS WEAVER AND DAN ELLIS

white worms don't work

Nicholas Weaver is a researcher at ICSI specializing in worms, computer security, and architectures for high-speed intrusion detection.

nweaver@icsi.berkeley.edu

Dan Ellis recently finished his Ph.D. at George Mason University. At MITRE, as an infosec scientist, he's been working on various aspects of worm technology since 2001.

ellisd@mitre.org

SEVERAL VOICES HAVE REPEATEDLY proposed using worms as a tool for fighting other worms, updating systems, and other security tasks. Such *white worms* (also known as anti-worms [2,5,17], predators [6], or nematodes [1]) have been proposed as a mechanism to counter a spreading worm. The idea is to launch a worm that spreads to patch the target and make it immune to the competing worm. Likewise, there have been several allegedly white worms both written (e.g., Code Green [7]) and released (e.g., Welchia [16] and Anti-Santy Worm [8]) with the alleged goal of cleaning up an existing infection.

Using white worms to immunize from or clean up after a competing worm is a bad idea for several reasons. Technically, writing worms to do what is desired is incredibly difficult. It is difficult to target a white worm to infect and immunize all of the machines of interest and *only* those machines. Also, getting the worm payload to do exactly what is desired is hard to do with significant testing, let alone on the fly in response to another worm. Consequently, the risk of the worst-case, compounded problem (deploying a damaging payload across a large number of hosts that do not belong to you) is practically impossible to mitigate. Even using a highly controlled worm for penetration testing on a live network [1] has serious issues.

There are also some significant legal reasons not to use a white worm. For example, infecting a machine that one does not own is a serious crime in most nations, independent of the intentions of the author of the white worm. Worse, if there is a problem in the targeting of the worm and the payload of the worm, significant damage is possible, which could even lead to international conflict if spread occurs across national boundaries, independent of intent or sponsorship [18].

Fortunately, a combination of machine attestation, automated inventory management, and patch management offers a superior and acceptable solution. A combination of these tools provides mechanisms to distribute patches and protective instructions faster than all existing worms. Even better, the fundamental time scale for patch distribution can equal or even exceed the fastest worm theoretically possible. Any host that could be configured in any way to improve or facilitate the performance of a white worm could also be instrumented with a patch-management system. We conclude that, without exception, the use of white worms poses an unacceptable risk.

What Is a White Worm?

A *white worm*, in our definition, is any worm released with allegedly benign intent. The objective could be to patch systems before a malicious worm is released, to out-compete an already spreading worm, or to clean up after a worm attack. We use the term *white* rather than *good* because the actions of such a worm may not be deemed beneficial by the owner of a targeted machine.

In all these scenarios, we assume that the white worm is spreading by exploiting some vulnerability in a target system, rather than by legitimately accessing the target machine (e.g., via an authorization daemon installed on the target). This is simply because if an authorized backdoor or daemon can be installed on the target computers, then a patch-management system can be deployed instead.

A patch-management system, in contrast to a white worm, is a purely consensual mechanism. All participating systems contain a small daemon that is used to receive and apply defensive instructions, including patches and network filters. Combined with machine attestation systems such as Cisco's Network Admission Control [3], this system can ensure that all systems within a well-structured corporate network are participating and able to receive and process updates.

Displacing an Existing Worm

The most commonly proposed application for a white worm is to displace a malicious worm already in place. Yet it is currently considered standard procedure for an attacker to close the vulnerability used to exploit the system, and we have heard anecdotal reports of even other unrelated vulnerabilities being patched. Compromised hosts are a resource to the attacker; thus, an attacker benefits from preventing that resource from being claimed by others.

As a specific example, Welchia [16] was called a white worm, because it removed a competing worm (Blaster) and installed the patch. Yet, Welchia also contained a malicious payload, opening up a backdoor on infected systems. Welchia's patching was really to prevent double-infection and to remove a competitor that made infected systems unstable.

Displacing a rootkit or worm that takes measures to defend itself can prove highly difficult. Since a worm author is likely to be less concerned about the stability of other applications on the target host, there is little motivation for a defensive worm author not to produce a patch suite. Whereas a conscientious sysadmin would vet (which may include testing negative impacts) a patch before applying it, a worm author has little motivation. Therefore a worm author may decide to patch other vulnerabilities not previously addressed by the sysadmin. As a white worm would be constructed primarily using publicly available exploits, it would be very difficult to successfully infect a previously infected and patched host.

A worm author may get the benefit of limiting access without patching if infection results in the host becoming invulnerable, which often occurs when the target application is single-threaded and the exploit never returns control flow to the victim program. Blaster on Windows 2000 [15], Witty [11], and Slammer [10] all showed this behavior. Additionally, the worm author could simply terminate the vulnerable service after infection, unless the worm author needs to use the vulnerable service or the service is essential to system operation.

Thus, unless there is a *separate* vulnerability known by the white-worm author but not by the original worm author, there is a vulnerability in the original worm, or the original worm author simply doesn't care about being displaced, a white worm can't be reliably used to displace an existing infestation.

Of course, the same problem applies to patch management. It is common for malicious code to disable antivirus and other defensive applications in the process of rootkitting a system. Thus, although a white worm cannot reliably displace a worm, neither can patch management. But this is part of the general problem of recovering compromised machines, not a specific limitation of patch management.

Outracing a Spreading Worm

Thus if a white worm is to out-compete a worm that is currently spreading, it must be created dynamically and released very quickly in response [2]. The problem is that if both worms are equally optimized, the worm with the head start has an effectively insurmountable advantage. Only if the anti-worm gets an unmatched performance boost through some other technique (e.g., hitlisting) can it be expected to compete with the initial worm.

However, the author of a malicious worm can use these same techniques to gain a speed advantage, obviating any optimizing effect of the white worm. Thus, it is best to consider the two worms, the initial malicious worm and the white worm, as having the same speed, as any speed improvements that the white-worm author could use could be copied by the author of the malicious worm.

We conducted a simple simulation using a modified version of the simulator from [14]. At $T = 0$, a malicious worm is released (300,000 vulnerable systems, 200 scans/s/worm, 32-bit IPv4 address space). After some specified fraction of the hosts are infected (0.01%, 0.1%, 1%, and 10%), an equivalent white worm is released from a single source and spreads at the same rate as the initial worm. For all but the 0.01% sensitivity, the white worm was effectively useless, having a negligible impact on the black worm's propagation and final infection. But even the supersensitive white worm, released when 0.01% of the systems are infected by the black worm, is almost useless, inoculating less than 3% of the victims in the median case. We conducted 100 simulations and graphed the median case, the case representing the top 5%, and the case representing the bottom 5% in terms of effectiveness. This is shown in Figure 1.

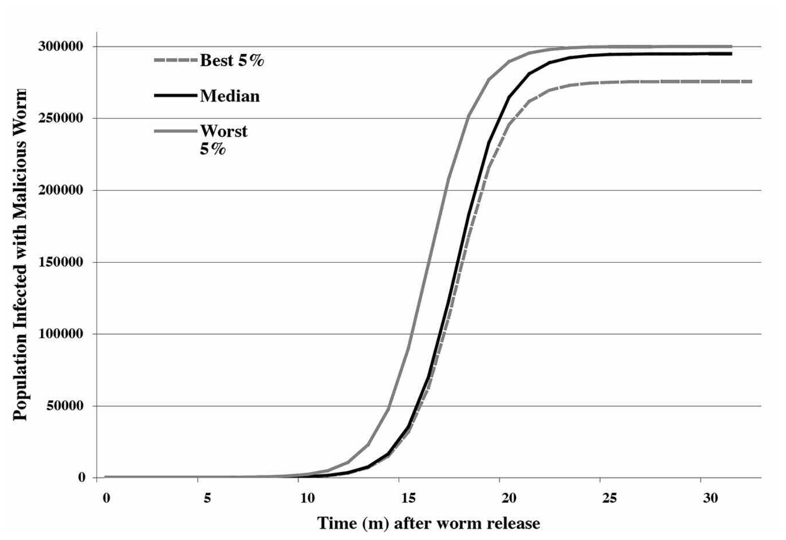

FIGURE 1: THE NUMBER OF SYSTEMS INFECTED BY THE BLACK WORM FOR THREE DIFFERENT SIMULATION RUNS (OUT OF 100), WHEN THE WHITE WORM IS RELEASED AFTER 0.01% OF THE VULNERABLE SYSTEMS ARE INFECTED. THERE ARE 300,000 VULNERABLE SERVERS.

Even with a supersensitive white worm, released after 0.01% of the systems are infected, most of the vulnerable population is still compromised by the malicious worm. This is because even with the highly (even optimistically) sensitive value of 0.01%, the black worm has infected 30 systems, giving it a nearly 5-generation head start over the white worm. If we even consider a perfectly timed white worm, equally matched to the malicious worm and released at the same time, the white worm is still expected to save only 50% of the systems.

Another important aspect of relying on a white-worm-based defense is the accuracy of the underlying detection mechanism. It is understood in the worm community that accuracy is the most important issue in developing a successful worm-detection capability [19]. The cost of a false alarm varies depending on the response. The best false alarm rates reported are on the order of just over once per day [4,13]. A false alarm that causes a significantly disruptive event, such as a premade white worm, could be catastrophic.

Disruption from a Worm

One problem with worms is they can be disruptive to the network. For example, an anti-Slammer, released at the same time as Slammer, would have proved equally disruptive (at least until the white worm stopped spreading). In fact, Welchia, owing to a faulty ICMP scanner, was far more disruptive to the network than Blaster, the worm it was displacing. Blaster's traffic was simply normal TCP, and its scanning rate was low. Welchia had a high-rate ICMP scanner that flooded the local network with unresponsive traffic that caused congestion, and it was Welchia (the alleged white worm), not Blaster, that disrupted the Navy/Marine Corps intranet [9].

Of course, even if a white worm is released before a malicious worm, the white worm may prove to be as bad as or worse than the disease. Assume that the patch deployed by the white worm requires a system reset, but the original worm does not affect system stability. In this case, the white worm's act of resetting a critical but infected system may be more damaging than an unchecked infection by the malicious worm.

Additionally, a single-packet UDP worm such as Slammer or Witty will naturally be disruptive to the network. It would require significant engineering to develop a congestion-sensitive UDP-based worm, and such a worm would be considerably slower than a malicious worm, which doesn't care about fair congestion control.

The Difficulty of Control

Even if a white worm is to be used for another task, such as proactively patching before a malicious worm can spread, there is still the substantial difficulty of controlling the system.

If the white-worm author doesn't a priori know which systems should be compromised and have provable coverage only over those systems, an error in this logic could prove catastrophic. The original worm paper by Shock and Hupp [12] only touched on the real limitations: the havoc caused by bugs in the worm that either would cause self-propagation (their model was mobile without duplication) or could disrupt the entire network.

The one proposal that attempts to address this, Nematodes, postulates either an end-host *allowed* token or an authorization server (in either case patch management could be deployed instead with much lower risk), or a *white graph* of allowed traversal topology.

But what happens if there is a failure (i.e., bug or unforeseen circumstance) in the white graph? For example, the Nematodes proposal remarks that attacking 192.169/16 space should be OK if the worm is already in 192.169/16 space (thus the notion of a white graph instead of a list). Yet what happens if happenstance causes a nematode to end up on a critical system in someone else's private address space (e.g., spreading via a laptop that changes locations and IP addresses)?

This is far worse than the problem of a bad input into a vulnerability scanner. A nematode, by infecting a host, may disrupt the host considerably more than a vulnerability scanner would. Worse, because of the self-propagating nature, a badly targeted nematode could spread to many more locations. With a bad entry in a vulnerability scanner, only those hosts affected by the entry will be discovered. But a bad entry in a nematode's target database could propagate, affecting far more systems.

The Legal Minefield

It is this difficulty of control that brings up the greatest problem. If the white worm's author has legitimate access and control of the systems, he or she can use a patch-management tool. But if the author does not have legitimate access and control of the systems the white worm ends up infecting, the worm's author or releaser just committed a crime (a felony in the United States).

Even a nationally authorized white worm could encounter these difficulties. If the worm itself infected a computer in a different country, the laws of the computer's location, not of the worm author's, apply. If the breach of the white worm or the payload is damaging enough, the victim nation may perceive it to be an act of war and retaliate commensurately [18]. The potential for catastrophic political damage, even for a governmentally authorized worm, is difficult to understate.

The Alternative: Patch Management

For networks that are controlled, patch management and system attestation provide all the benefits of a white worm with a much lower risk profile. System attestation (such as Cisco's NAC [3]) prevents any system from connecting that is not running a set of administratively mandated software, such as antivirus and patch-management tools. A patch-management system allows the administrator to push code to the systems dynamically. Using these two capabilities together, it is reasonable to assert that hosts are attested and up-to-date with the most current patches, assuming they haven't already been compromised and rootkitted.

Unlike a white worm, attestation and patch management has perfect and controlled coverage: All systems on which the patch-management software is installed, and *only* those systems, are updated. There are no issues with self-propagating code escaping into the wild. Patch-management systems are much easier to test than a white worm, as their behavior is more deterministic. Finally, patch-management systems can be intrinsically faster than a worm, as there is no target discovery, TCP sessions can be preestablished, and, with multicast, the same data can be sent to all systems simultaneously.

Patch management, however, is not without cost. It may require additional software, testing infrastructure, and sitewide policies to deploy. However, a well-run enterprise already needs a patch-management system to handle the large number of upgrades and patches for a variety of software packages. It seems far more reasonable to invest in extending that capability than it is to invest in a technology whose potential negative impacts are practically impossible to bound.

Conclusions

The theme of using white worms has recurred roughly annually over the past few years as one proposal to deal with network worms. Network worms are a serious threat because they can spread quickly and deploy an arbitrary payload. It is natural to want to harness that power to do something useful. One naive proposal is to use them to counter other worms. However, the potential negative consequences of a white worm solution are exceptionally grave, because they cannot be practically bounded. Further, a patch-management system out-competes white worms in terms of performance (speed and coverage) and has a far more acceptable risk profile. The capabilities are also technically more mature and cost-effective than those of white worms. We, therefore, call on the community to discourage the consideration of white worms and focus intellectual effort on the many other hard problems in computer security.

REFERENCES

[1] D. Aitel. "Nematodes—Beneficial Worms": http://www.immunityinc.com/downloads/nematodes.pdf.

[2] F. Castaneda, E. C. Sezer, and J. Xu. "Worm vs. Worm: Preliminary Study of an Active Counter-Attack Mechanism," *Workshop on Rapid Malcode (WORM)*, 2004.

[3] Cisco network admission control: http://www.cisco.com/en/US/netsol/ns466/networking_solutions_package.html.