

# ;login:

THE MAGAZINE OF USENIX & SAGE

August 2003 • volume 28 • number 4

## inside:

**SYSADMIN**

Geer: Patch Work

**USENIX & SAGE**

The Advanced Computing Systems Association &  
The System Administrators Guild

# patch work

Absolutely no one in this audience needs an introduction to patching. It's a pervasive requirement as well as a pervasive nuisance, and that's putting it mildly. If it weren't such a nightmare it wouldn't be as interesting, and vice versa.

This is not a security article. For a mature environment, security is a subset of reliability; in other words, this is a reliability article. The logic is thus:

If a system is insecure, then  
    It is unreliable, therefore  
        Security is necessary for reliability, yet  
            Security is insufficient for reliability, therefore  
                Security is a subset of reliability.

The axis of evaluation of a single patch or a system for patching or a regime for the management of patches is one that begs for the metrics of reliability. Reliability metrics are well defined in other fields, so just steal them fair and square. While we're at it, let's not ignore business reality – just as public lotteries are a tax on those who cannot do math, a patch is a tax on those, upstream or down, who make software reliability something that can only be bought on the installment plan.

What, then, is the “physics” of patching? First: The more serious the hole that the patch is to fill, the less time you have to fill it. This is especially true for security patches, because security patches come with the shortest fuses, already lit. For those who actually know physics, this might be our equivalent of “inflation.”

Second: Patch density is itself a function of risk over space and time. It is a function over space since no automatic system ever works completely, hence shoe leather is still required, which breaks the first rule of scalability for distributed systems: No shoe leather. It is a function over time as exploitability increases monotonically once a flaw has been discovered and one must always assume that a flaw that is officially known was discovered unofficially much earlier, i.e., the countdown clock is already running by the time you realize that there is something to patch.

Third: Patches are the high-energy radiation around the black hole of a desktop monoculture. The more massive the black hole, the higher the energy of that radiation. The gravity of the desktop monoculture in particular absorbs such a huge mass of the slings and arrows of outrageous fortune seekers (the exploit writers) that it radiates particularly high energy photons (patches whose terminal velocity is proportional to the curve of susceptibility, which in true platform monocultures is a singularity).

Fourth: Critical mass – it's not just for weapons anymore. All the failures that actually matter are cascade failures. Cascade failure, like an epidemic or a chain reaction, is a big-number interaction of virulence (radioactivity) and immunity (isotope stability). What may matter most to you is not whether you patch but whether your communicating counterparties patch. If they are all sending you a tsunami of Slammer, it might not actually matter if you are patched or not. Your susceptibility to the clumsiness of others is the other half of Metcalfe's Law, the half with the sign bit reversed. In the sense of critical mass, “we” are all in this together even if it approximates roping together amateur mountain climbers.

If you're losing a game, first try to change the rules. What might that mean here? For starters, we might consider source reduction, delivery automation, mitigation by interdiction, and risk transfer.

Source reduction means what it sounds like, and it is easier to say than to do. Let's be abundantly clear: This is a reliability matter related to trying to cram 10 pounds of

by Dan Geer

Dan Geer is a USENIX Past President and is Chief Technology Officer at @Stake, Inc.



geer@atstake.com

To err is human; to really foul things up you need computers.

functionality into a five-pound bag, and the only way we'll get more reliable software is to raise prices for software. The richest software vendor in the world mostly doesn't care how much it costs to cure quality flaws so long as it doesn't cost time to market, but source reduction can come as a side effect of liability judgments against suppliers, directly gated by quality metrics imposed by energized buyers, or the market pricing impacts of differential insurance premiums driven by payouts. Somebody will pay; if not, you call it risk transfer.

Delivery automation, also known as automatic update, takes a tough problem and makes it brittle. So long as the automatic update works it is a solid win. When it fails, whether by incompetence or treachery, it demonstrates that while to err is human, to really foul things up you need computers. Nevertheless, it is in our future by contract even if not by technical evaluation. Absent real emergencies, pushed patch notifications with lazy evaluation by the notified client recommends itself, especially if coupled with . . .

Mitigation by interdiction has scaling factors that strongly recommend it if, and this is a big if, there is a perimeter at which to apply that interdiction. Removing toxic attachments from email is a kind of patching – not that the Devil didn't have a backslappingly good day when some clown decided that shoe-horning executables into text messages sounded cool – and “we” think nothing of automated patching of email these days just as I'm sure we'll soon think nothing of patching instant messages, SMS opacities, and even the stuff that comes from P2P “networks.”

But let's get back to the net-net of patching with respect to reliability, which is what this is about. As Mike O'Dell used to say, left to themselves creative engineers will deliver the most complex system they think they can debug. He was talking about the counterweight of having enough old hands around whose “sadder but wiser” experience balanced out “what could go wrong?”; but I'm thinking about complexity and the debuggability margin as where complexity and reliability meet. There's no doubt that reliability and complexity have a difficult marriage where each has caused the other pain. Reliability can enable complexity and complexity can enable reliability, but creativity-fueled high rates of change generate the kind of complexity that is decoupled from reliability. That can be good in a greenfield startup; it can be bad in the long distance call switching system.

The needed maturity of the area of application determines the rate of change that can be tolerated, which brings me to a central point: If you think of a patch as a software release, then you divide the world of software releases into the voluntary and the involuntary from your, the end-using customer's, point of view. That, dear friends, allows you to use the ratio of involuntary to voluntary software releases to just flatly rank software vendors. You know what I am thinking, but this is a family magazine.

There are not enough of us (USENIX) to go around, so there has to be substantial automation of patch delivery or a substantial reduction in patch necessity. If the number of people with Internet access is more-or-less doubling every six months, then only one in a million was here when Mosaic first appeared. That's boring, but curves like that absolutely tell you that automation is going to have to substitute for “been there, done that” levels of experience among the great mass of the patch-needy. If what obtains is that there's neither automation nor enough of “us” to go around, then the inevitable injuries to “the innocent” guarantee that the liability law system will take over and genuine innovation will get harder to fund and deploy.

This is bad news, overly simplistic, unignorable. Those of you who can measure things – please do some measurement and publish your numbers. Numbers get believed. Bellyaching gets a cool compress. Clock's ticking.